Project Acronym:	PhasmaFOOD
Grant Agreement number:	732541 (H2020-ICT-2016-1 - RIA)
Project Full Title:	Portable photonic miniaturised smart system for on-the-spot food quality sensing

DELIVERABLE

Deliverable Number	**D4.1**
Deliverable Name	**First implementation of software for PhasmaFOOD smart food analysis system**
Dissemination level	PUBLIC
Type of Document	REPORT
Contractual date of delivery	M17
Deliverable Leader	WINGS
Status & version	V2.0
WP / Task responsible	WP4 / T4.1, T4.2
Keywords:	Software architecture, mobile application, cloud platform, communication APIs, database, development tools, data model, dashboard, data analysis, security, privacy
Abstract:	This deliverable reports on the first implementation of the software for the PhasmaFOOD smart system. This deliverable reports on the implementation of the building blocks distributed between the embedded software, mobile application and cloud platform. The document specifies communication APIs, database organization, data analysis, decision making mechanisms, security and privacy features, and interfaces with end users.

This document is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732541. It is the property of the PhasmaFOOD consortium and shall not be distributed or reproduced without the formal approval of the PhasmaFOOD Management Committee.

Deliverable Leader:	Paraskevas Bourgos (WINGS)
Contributors:	Milenko Tosic (VLF), Ioannis Daskalopoulos (INTRA), George Tsoumanis (INTRA), Eugenio Martinelli (UTOV), Benedikt Groß (FUB), Konstantinos Tsoumanis (WINGS), Aimilia Badouna (WINGS), Aikaterini Demesticha (WINGS), Panagiotis Demestichas (WINGS), Nelly Giannopoulou (WINGS), Dimitris Kelaidonis (WINGS), Orestis-Andreas Liakopoulos (WINGS), Ioannis Maistros (WINGS), George Makantasis (WINGS), Konstantinos Petsas (WINGS), Aspa Skalidi (WINGS), Vera Stavroulaki (WINGS), Ioannis Stenos (WINGS), Konstantinos Trichias (WINGS), Konstantinos Tsagkaris (WINGS), Giannis Tzanetis (WINGS), Evaggelia Tzifa (WINGS), Panagiotis Vlaheas (WINGS), Apostolos Voulkidis (WINGS)
Reviewers:	Benedikt Groß (FUB), Susanne Hintschich (IPMS)
Approved by:	George Koutalieris (INTRA)

Executive Summary

The main goal of this deliverable is to report on the implementation of the PhasmaFOOD software architecture on functional and system level. The main building blocks implemented for embedded software (on the PhasmaFOOD sensing device), mobile application and the cloud platform are reported. The document focuses on the PhasmaFOOD mobile application as integral part of the software architecture. The implementation details of the mobile application software in data preprocessing, decision making, process configuration and interaction with the end user is described. Then, the deliverable moves onto the PhasmaFOOD cloud platform as the focal point for all contextual and sensory data collected by the PhasmaFOOD sensing device and the mobile application. This document presents the main building blocks and functionalities of the PhasmaFOOD cloud platform and its interactions with the rest of the system architecture. Also, a preliminary report on the data analysis and decision making processes to be used in the cloud platform is presented. Finally, the document provides an overview of security and privacy protection procedures that will guide software development. At the end the deliverable describes the PhasmaFOOD SW development strategy with the selection of languages, tools, frameworks and libraries that are utilized for the software development.

Table of Contents

Executive Summary 3
Definitions, Acronyms and Abbreviations 9
1 Overview of the PhasmaFOOD software architecture 12
1.1 Embedded software 14
1.2 Mobile application 14
1.3 Cloud platform 15
1.4 Matching requirements to software architecture 16
2 Data analysis and decision making 16
2.1 Data analysis algorithms 16
2.2 Decision making algorithms 20
2.3 Matching requirements to data analysis and decision making algorithms 23
3 PhasmaFOOD mobile application 24
3.1 Mobile application architecture 24
3.1.1 Block-level description 24
3.1.2 Usage UML diagram and description 26
3.1.3 GUI and UX 28
3.2 Communication with sensing device 31
3.2.1 Description 31
3.2.2 APIs and API handlers 32
3.3 Communication with cloud platform 34
3.3.1 Description 34
3.3.2 APIs and API handlers 34
3.4 Matching requirements to mobile application development 36
4 PhasmaFOOD cloud platform 36
4.1 Cloud platform architecture 36
4.1.1 Block-level description 37
4.1.2 Data model for collecting laboratory data 41
4.1.3 Structure of the cloud database 43
4.1.4 Hosting platform description 44

4.1.5 Dashboard description 45
4.2 Cloud data analysis and machine learning models 49
4.2.1 Access to the training database 50
4.2.2 Defining machine learning models 51
4.2.3 Model training and storing 51
4.3 APIs to 3rd party databases 53
4.4 Matching requirements to cloud platform development 53
5 Security and privacy 55
5.1 User registration and authorization 55
5.2 Device registration and authorization 56
5.3 Trusted data sources 56
5.4 Privacy protection 57
5.5 Matching requirements to data security and privacy 57
6 Software Development strategy 58
6.1 Languages, tools and frameworks used for development 58
6.2 Open source libraries 60
6.3 Validation plan 61
6.3.1 Training and test data sets 61
6.3.2 Training of machine learning models on the cloud platform 62
6.3.3 Testing of communication flow 62
6.3.4 Testing of communication interfaces 63
6.3.5 Testing of decision making algorithms 63
7 Conclusion and next steps 64
8 References 65
Appendix A 67
8.1 Data Model (JSON Schema Format) 67

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

Document History			
Version	**Date**	**Contributor(s)**	**Description**
V0.1	08/03/2018	WINGS	ToC
V0.2	24/04/2018	WINGS, VLF, FUB, INTRA	Final ToC, Assignment to partners
V1.0	26/05/2018	WINGS, VLF, FUB, INTRA, UTOV	First complete version
V1.1	29/5/2018	FUB, WINGS, INTRA, VLF	Updates based on FUB's review
V2.0	31/5/2018	IPMS, VLF, WINGS	Updates based on IPMS' review

Table of Figures

Figure 1-PhasmaFOOD SW architecture overview 13
Figure 2 PhasmaFOOD embedded SW high level structural diagram 14
Figure 3 A sketch of Principal Component procedure. PCA finds in the dataset the direction with maximum variance 17
Figure 4 A sketch of the Fisher Discriminant Analysis 20
Figure 5 - PhasmaFOOD reactive decision making mechanism....22
Figure 6 - PhasmaFOOD proactive decision making mechanism 23
Figure 7 - PhasmaFOOD mobile application block diagram 24
Figure 8 - MVVM typical workflow 26
Figure 9 - UML usage diagram of the PhasmaFOOD mobile application....27
Figure 10 - Mobile application GUI - Hoe screen and device pairing 29
Figure 11 - Mobile application GUI - measurement configuration wizard (Use case 1 – Mycotoxin detection example)...30
Figure 12 - Mobile application GUI - configuration and individual sensors, measurement data and data analysis result presentation....31
Figure 13 - UML sequence diagram of the embedded-mobile API 32
Figure 14 - UML sequence diagram of the cloud-mobile API 36
Figure 15 - Block diagram of the PhasmaFOOD cloud platform 37
Figure 16 - Firebase Cloud Messaging mechanism 41
Figure 17 - Basic operation of the ZeroMQ publish-subscribe mechanism 41
Figure 18 - PhasmaFOOD cloud database 44
Figure 19 - UML usage diagram of the PhasmaFOOD web dashboard....47
Figure 20 - Web dashboard - landing page (login)....48
Figure 21 - Web dashboard - preliminary approach for presenting individual measurement data 49
Figure 22 - JWT token based authentication and authorization 56

List of Tables

TABLE 1 - PHASMAFOOD MOBILE APPLICATION BUILDING BLOCKS 24
TABLE 2 - THE MAIN BUILDING BLOCKS OF THE PHASMAFOOD CLOUD PLATFORM 37
TABLE 3 - DEVELOPMENT LANGUAGES, TOOLS AND FRAMEWORKS USED FOR IMPLEMENTATION OF THE FIRST PROTOTYPE OF THE PHASMAFOOD SW PLATFORM 58
TABLE 4 - OPEN SOURCE LIBRARIES FOR REALIZATION OF THE FIRST PHASMAFOOD SOFTWARE PLATFORM PROTOTYPE 60

Definitions, Acronyms and Abbreviations

Acronym	Title
3GPP	3rd Generation Partnership Project
AAA	Authentication, Authorization and Accounting
AES	Advanced Encryption Standard
AF	Aflatoxin
AMP	Alternate MAC/PHY
ANN	Artificial Neural Network
API	Application Programming Interface
APN	Apple Push Notification
ATT	ATTribute protocol
AWS	Amazon Web Services
BCN	Battery Charging Needed
BR	Basic Rate
BT	Bluetooth
BT LE	Bluetooth Low Energy
CAC	Channel Access Code
CMOS	Complementary Metal-Oxide Semiconductor
CPHA	Clock Phase
CPOL	Clock Polarity
CPU	Central Processing Unit
CO	Confidential, only for members of the consortium (including Commission Services)
COTS	Commercial Of The Shelf
CSS	Cascading Style Sheets
CRC	Cyclic Redundancy Check
D	Demonstrator
DA	Data Analysis
DB	Database
DCT	Discrete Cosine Transform
DDR3	Double Data Rate Type 3
DL	Deliverable Leader
DRAM	Dynamic Random-Access Memory
DWT	Discrete Wavelet Transform
Dx.x	Deliverable (where x defines the deliverable identification number e.g. D1.1.1)
EDR	Enhanced Data Rate
EU	European Union
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GCM	Google Cloud Messaging
GTS	Go To Sleep

GUI	Graphical User Interface
HOD	High level of Oscillations Detected
HTD	High Temperature Detected
HTML	Hypertext Markup Language
HTTPS	Hyper Text Transfer Protocol Secure
HW	Hardware
I2C	Inter-Integrated Circuit
IC	Integrated Circuit
IMU	Inertial Measurement Unit
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
K-NN	K - Nearest Neighbors
KPI	Key Performance Indicator
LBD	Low Battery Detected
LDA	Linear Discriminant Analysis
LED	Light Emitting Diode
LOD	Limit of Detection
MAC	Media Access Control
MCU	Microcontroller Unit
MIC	Message Integrity Check
MISO	Master Input Slave Output
ML	Machine Learning
MMU	Memory Management Unit
MOSI	Master Output Slave Input
MPU	Microprocessor Unit
MQTT	Message Queue Telemetry Transport
MSC	Multiplicative Signal Correction
NCA	No Connection Achieved
NIR	Near Infrared
OS	Operating System
PAL	Protocol Adaptation Layer
PCA	Principal Component Analysis
PDU	Protocol Data Unit
PHY	Physical
PLS	Partial Least Squares
PLSDA	Partial Least Square Discriminant Analysis
PMIC	Power Management unit Integrated Circuit
POR	Preprocessing Operations Required
PU	Public
R	Report
RCN	ReConfiguration Needed

RFCOMM	Radio Frequency COMMunications
RGB	Red Green Blue
REST	Representational State Transfer
RPP	Restore Previous Process
RSD	Read Sensors Data
SCLK	Serial CLocK
SDA	Serial Data
SDK	Software Development Kit
SiP	System in Package
SNV	Standard Normal Variate
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SS	Slave Select
SSL	Secure Sockets Layer
SVM	Support Vector Machine
SUP	Start UPdate
SW	Software
TCP	Transmission Control Protocol
TCS	Telephony Control Specification
TLS	Transport Layer Security
TSD	Transmit Sensory Data
UI	User Interface
UML	Universal Markup Language
UON	User pressed button ON
UPB	User Pressed Button
UPC	UPdate Completed
USB	Universal Serial Bus
UV	Ultraviolet
UX	User Experience
VIS	Visible
WP	Work Package

1 Overview of the PhasmaFOOD software architecture

The PhasmaFOOD system comprises three distinctive layers integrated together to provide scalable sensory and contextual data collection, as well as processing and analysis for food analysis related decision making. These three layers include: a portable sensing device, an end user mobile device (i.e. smartphone and tablet) and a cloud platform. The PhasmaFOOD software architecture is defined with the goal of taking full advantage of the envisioned system architecture. On the sensing device level, the embedded software integrates physical sensors including their configuration and read-out, data preprocessing, control logic and a communication interface. The PhasmaFOOD mobile application provides interface between the portable sensing device, as the primary source of sensory data, and the cloud platform which is the primary sink for all data streams. The mobile app also provides the interface towards the end user for configuring the food analysis process/measurements as well as visualizing and interpreting analysis results. The PhasmaFOOD mobile application may perform additional data processing in order to ensure that only relevant data is delivered to the cloud platform. Finally, the PhasmaFOOD cloud platform is the focal point of all sensory and contextual data coming from the portable sensing device, the mobile app and, if necessary, 3rd party data sources. The cloud platform is where the collected data is analyzed in depth. The platform hosts a data warehouse for all sensory and contextual data sets necessary for performing food analysis and deriving results. From this data warehouse data marts are derived for training different Machine Learning (ML)/Data Analysis (DA) models for realization of project use cases. Decision making algorithms (based on the trained ML models) are hosted on the cloud platform as well. Reactive (based on end user measurements) and proactive (strategic and based on data collected over time) decision making is performed based on the results of data analysis. Figure 1 shows a high level overview of the PhasmaFOOD software architecture.

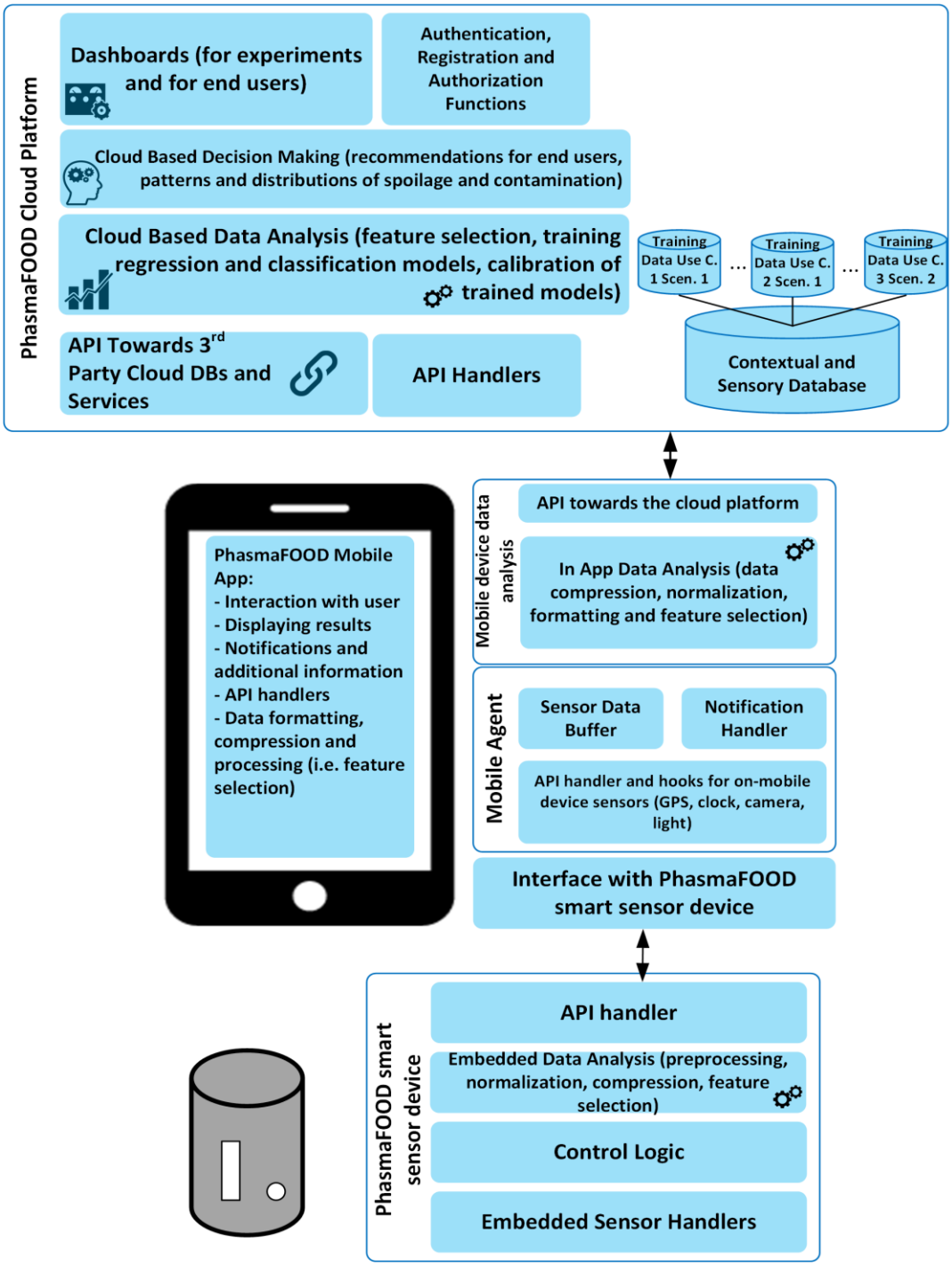

Figure 1-PhasmaFOOD SW architecture overview

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

1.1 Embedded software

The embedded software runs on the PhasmaFOOD sensing device and directly interfaces with the physical sensors on one side and prepares data for use case centric analysis performed throughout the PhasmaFOOD system architecture. The key modules, comprising the PhasmaFOOD embedded SW, are presented in the Figure 2 below.

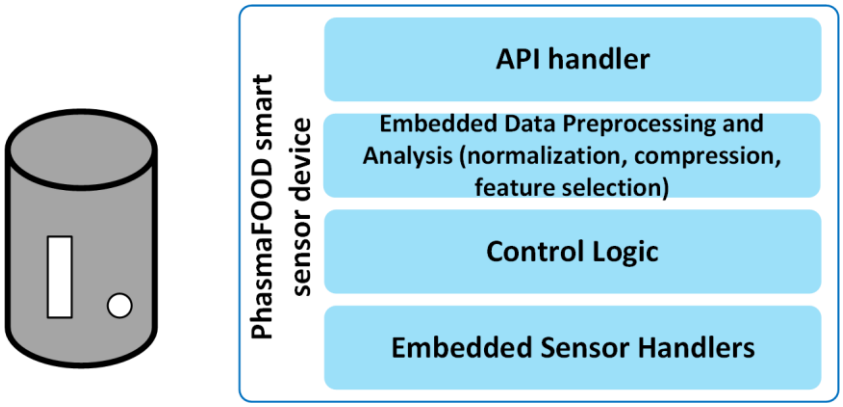

Figure 2 PhasmaFOOD embedded SW high level structural diagram

The PhasmaFOOD embedded software is responsible for the following key functionalities:

- To configure the sensors with the parameters for the required type of measurement.
- To run the activities of the sensors and light sources as a routine adjusted to the required type of measurement.
- To receive the raw measurement data generated by the sensing components (NIR, UV-VIS and camera).
- To pre-process and normalize the received measurement data. These functionalities include averaging, subtraction and division operations for the spectroscopy data, as well as compression and feature selection for the image data.
- To communicate the acquired measurement data to the mobile application and optionally directly to the cloud platform. The use of the latter functionality will be assessed to be integrated in the final PhasmaFOOD prototype.
- To optionally perform decision making based on trained ML models and implemented procedures for each or some PhasmaFOOD use cases. The integration of this functionality will be assessed for the final PhasmaFOOD prototype.
- To optimize the battery duration of the PhasmaFOOD device using power-saving modes.

More details about the embedded software are found in D2.1 [1] and D5.3 [2] deliverables.

1.2 Mobile application

PhasmaFOOD mobile application is the middle layer in the PhasmaFOOD SW platform hierarchy. It has three main roles in the SW architecture of the PhasmaFOOD system:

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

- To provide the communication bridge between the embedded system of the smart sensing device and the cloud platform hosting cloud DB and data analysis models. The mobile application establishes a communication channel with the embedded system and sensing device over Bluetooth interface for sending measurement configuration parameters and receiving measurement data and measurement process feedback. On the other side, the mobile application maintains API with the PhasmaFOOD cloud platform for user and device registration and authorization, requesting data analysis on prepared measurement data and receiving analysis results.
- To provide graphical user interface (GUI) towards the end users for managing PhasmaFOOD account, configuring individual measurements, configuring individual sensors, viewing measurement status and history and displaying food analysis results. The application also provides the end user with real time feedback on the measurement process (progress, alerts for correcting actions, current step in data communication and analysis).
- To, optionally, perform data preprocessing and decision making based on trained ML models and implemented procedures for each PhasmaFOOD use case.

The PhasmaFOOD mobile application will be developed for Android and iOS system. It will support both expert users (users in food analysis laboratories) and non-expert users who seek quick and simple interaction with the PhasmaFOOD system. For the first SW platform prototype we are focusing on development of the Android version (faster development and distribution) of the mobile application for expert users. Based on validation of the application by expert users both within and outside of the PhasmaFOOD project, we will specify a version of the application for non-expert users comprising a subset of functionalities enabled for expert users.

1.3 Cloud platform

The cloud platform is the focal point for all sensory and contextual data. The main roles of the PhasmaFOOD cloud platform are:

- It hosts the PhasmaFOOD cloud database and data warehouse. The cloud database is in fact two separate databases:
    - Data warehouse - this database is used for storing all datasets used for training decision making and data analysis models. It also stores trained ML models for all use cases and sub use cases.
    - Operational database – this database stores user profiles, unvalidated measurement data and analysis results.
- The cloud platform performs data analysis chains specifically tailored and optimized for each project sub use case. These chains comprise data formatting, feature selection and classification or regression procedures through which received measurement data are analyzed in order to inform the end user on food state.

- Reactive and proactive decision making procedures are hosted on the cloud platform. These mechanisms are rule engines and state machines, which decide on proper data handling and analysis chain, system calibration and updates as well as overall data flows through the system.
- The cloud platform hosts the cloud-mobile API handler and 3rd party API handlers for system integrators.
- The web dashboard is hosted on the cloud platform. It provides interface for end users and system administrators to see the datasets, measurement history, to try different data analysis chains on the same measurement data and specify new data set to be collected with the PhasmaFOOD system.

The PhasmaFOOD cloud platform in the first system prototype supports project partners in training food analysis algorithms, collecting and validating data sets for project use cases and validation decision making procedures. Based on these activities, the optimal set of procedures will be provided to the end users (experts and non-experts) in the final prototype.

1.4 Matching requirements to software architecture

Additional requirements' matching can be found in D5.3 on "Implementation of embedded software for PhasmaFOOD smart food analysis platform" Section 4.

2 Data analysis and decision making

2.1 Data analysis algorithms

In this paragraph a brief introduction on the techniques utilized in this preliminary phase for the analysis of data coming from different sensing elements (VIS, NIR spectrometer, etc.). The first one is the Principal Component Analysis (PCA) used as an unsupervised technique but also as a standard pre-processing technique. The Partial Least Square (PLS) is used as regression model and inear Discriminant Analysis (LDA) is applied for the classification problem. These linear techniques have been preferred because preliminary datasets were not large enough to guarantee a reliable result when non-linear techniques are used.

Before the description of data analysis techniques, a brief description of the preprocessing technique is given in the following section:

2.1 Preprocessing step

Before applying the regression or classification model, the data have to be pre-processed in order to reduce as much as possible all the non-reproducibility sources. For this reason, according to the sensing devices a different normalization is applied. For example, in the case of the VIS spectrophotometer a normalization using the reference and dark measurements is considered:

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

$$X'_{Sample} = \frac{X_{Sample} - X_{Dark}}{X_{Ref}}$$

More details about the data normalization are given in the deliverables D2.1 [1], D3.1 [3] and D5.3 [2].

2.2 Principal Component Analysis (PCA).

PCA is an unsupervised technique used to describe a dataset in a graphical representation carrying most of the data variance [4]. The axes of the new representation space are given by a linear combination of the original axes and they are requested to be uncorrelated and orthogonal. The first axis, called first "Principal Component", is chosen in the direction of the largest variance, the second axis, called second "Principal Component", is taken along the orthogonal direction to the first PC with largest variance, and so on. This procedure is depicted in Figure 3.

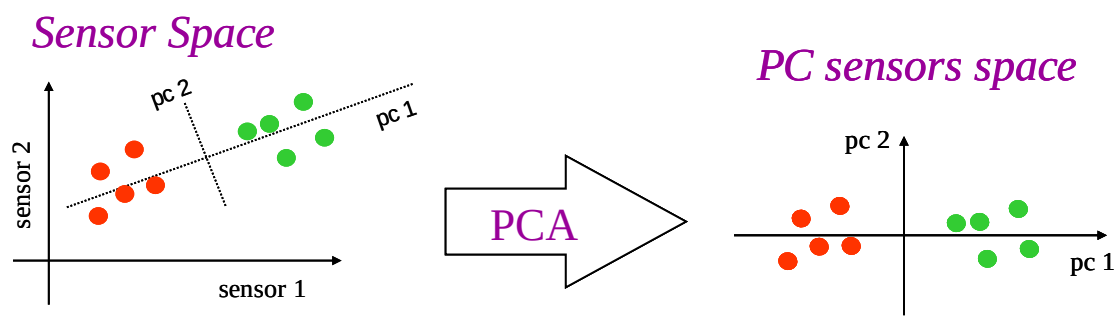

Figure 3 A sketch of Principal Component procedure. PCA finds in the dataset the direction with maximum variance

Principal Component Analysis is an algorithm that searches the optimal representation of the data in a subspace of smaller dimension, maximizing the variance of the dataset [4].

A scatter plot where the samples are represented as points in a space with coordinates the principal components (PCs) is called Scores Plot. Figure 3 (left) shows an example of Scores Plot in which with only a first PC it is possible to discriminate the two classes.

The plot of the features on the plane spanned by the PCs is called 'loadings plot' and shows the contribution of the features to the calculation of the PCs.

The loadings are concerned with the feature. They measure the contribution of each feature to the array. A high loading, for a feature, means that the principal component is aligned along the feature direction.

On the other hand, features with a similar behaviour in all the PCs brings the alike information.

In the data analysis for the different use-cases PCA is applied as pre-processing technique to reduce the data dimension used as input for the classification phase.

2.3 Partial Least Square (PLS).

Partial Least Square (PLS) regression is a data analysis technique that combines characteristics from principal component analysis and multiple regression. It is mostly useful when we need to predict a set of dependent variables from a large set of independent variables (predictors).

It originated in social sciences [4] but became popular in chemometrics due in part to Svante Wold [5] [6].

Let us consider M observations described by K dependent variables which are stored in a MxK matrix denoted Y, the values of J predictors collected on these M observations are collected in the MxJ matrix X.

The goal of PLS regression is to predict Y from X and to describe their common structure.

PLS regression finds components from X that are relevant for Y. PLS regression searches for a set of components, called latent vectors, that achieve a simultaneous decomposition of X and Y with the restriction that these components explain as much as possible of the covariance between X and Y. Then, the PLS algorithm searches the subset of X that shows the maximum correlation with Y and provides as final output a regression vector B. The regression vector B is used to predict from the input data the output variable Y_{est}.

More details about the PLS algorithm can be found in ref [5] [6].

It is noteworthy that PLS is a projection method. Then, this technique can be seen as a projection of the X-matrix down on a K-dimensional plane in such way that the coordinates of the projection (t_a a=1, …,K) are good predictors of Y.

The prediction error of the model is strongly dependent on the number of variables (features) and on the number of measures. In the case of small datasets, a cross validation technique is applied to obtain a realistic error estimation. The number of the Latent Variables that minimize the prediction error in the validation phase, are used to build a model for the further test phase.

2.3.1 Leave One Out Cross Validation (LOOCV).

In literature, several techniques of cross validation have been introduced, here the Leave One Out Cross Validation is considered [4].

Since the number of samples is rather poor, the prediction error rate is computed using cross-validation by blocks or a leave-one-out cross-validation; the data set is divided into k subsets: the calibration is carried out on (k-1) blocks, and the prediction is made on the samples belonging to the k-th subset. This is repeated k times with block permutation, in order to

predict all the samples. The cross-validation is called leave-one-out cross-validation where each subset is composed of only one sample: in this case, there are as many subsets and models as samples (k=n).

At the end of validation phase the Predicted Residual Error Sum of Squares is calculated in the following way:

$$PRESS = \sum_{i=1}^{N} (\hat{\boldsymbol{y}}_i - y_i)^2 \quad (7)$$

where $\boldsymbol{y}_i$ is the estimated value and y_i the real value.

The number of latent variables that shows a minimum of PRESS corresponds to the optimal choice. This kind of choice avoids model over fitting.

At the end of the validation procedure, chosen the *k* optimal latent variables, the Root Mean Square Error of Cross Validation (RMSECV) is calculated in the following way:

$$RMSECV = \sqrt{\frac{\sum_{i=1}^{N} (\hat{\boldsymbol{y}}_i - y_i)^2}{N}} \quad (8)$$

where $\boldsymbol{y}_i$ is the estimated value and y_i the real value.

This value gives a generalization of the prediction error of the model. This result is relevant to have an idea of the real prediction performance of the model in presence of data not used in the training phase.

2.4 Fisher Discriminant Analysis.

Linear Discriminant Analysis (LDA) or often called Fisher discriminant analysis, is the most common classification model used for pattern-classification and machine learning applications [7]. The goal is to project a dataset onto a lower-dimensional space with good class-separability to avoid overfitting and also to reduce computational costs.

The idea of the algorithm is summarized in the figure below, where the blue arrow indicates the best projection in term of class separability (see Figure 4).

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

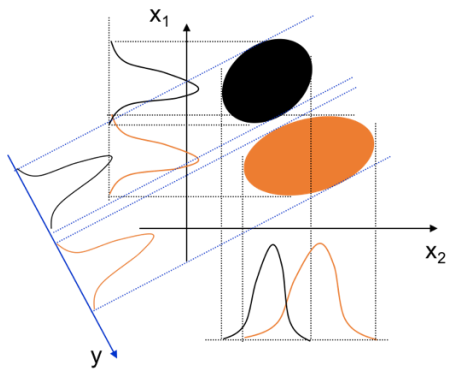

Figure 4 A sketch of the Fisher Discriminant Analysis

From the figure it is evident that the primary purpose of LDA is to separate samples of distinct groups by transforming them into a space (the blue line) which maximises their "between-class separability" while minimizing their "within-class variability".

2.2 Decision making algorithms

The PhasmaFOOD software platform provides reactive and proactive decision making based on collected data sets, provided measurements from end users and system defined triggers. These algorithms are the basis for realization of project use cases, system management and strategic planning of monitored supply and production chains. Decision making mechanisms are implemented as collections of software procedures and scripts which take as inputs trained ML models, the current measurement/trigger on which to conduct analysis and employ rule engine to execute all the steps leading to the final decision, whether it is a reactive decision or a strategic/proactive decision.

In the first prototype of the PhasmaFOOD software platform, the decision making mechanisms are distributed between the embedded system and the cloud platform. The embedded system employs reactive decision making in form of a state machine. This state machine is responsible for employing embedded system resources to collect measurement data based on the measurement configuration provided by the end user:

1. Measurement configuration comprises selection of measurement use case, sample type and sample conditions (specific to project use case and sample combination). Based on these configuration parameters the embedded system will know which sensing sequence to employ (combination of spectrometers, camera, illumination and time) and which preprocessing algorithms to use (normalization, compression, white reference division, dark reference subtraction). The decision making procedures and sequences for properly utilizing embedded system sensing unit and properly handling obtained data is still work in progress and preliminary approach will be reported in D6.1.
2. Also, the embedded system is able to receive configuration and calibration parameters from the mobile application, through a direct specification by the end user. This is

required mainly during the setup phase, while optimizing the sensor and lighting configuration for a certain use case. Based on these parameters the embedded system knows whether to utilize default calibration values for spectrometers and the camera – defined for a specific project use case/ dataset. If the end user chooses to change calibration parameters, the embedded system through driver software of each data source (spectrometers and camera) can change the parameter value and collect measurements with new set of configuration parameters. Acceptance of the new measurement data is performed manually by the end user. The end user is presented with a chart (value + wavelength) for NIR and UV-VIS spectrometers of the new measurement. Based on visual inspection, experienced users can quickly identify erroneous measurements and discard them. The control state machine of the embedded system is described in detail in deliverable D5.3 [2]. Other decision making procedures are implemented on the PhasmaFOOD cloud platform.

Figure 5 shows a high level block diagram of the reactive decision making process performed on the PhasmaFOOD cloud platform. There are two main functionalities of the reactive decision making:

1. Deciding, which data analysis and machine learning pipeline to utilize on the provided measurement data. This decision is based on measurement configuration parameters provided by the end user through the mobile application. The configuration parameters are specific for each integrated sensor (NIR, VIS and camera) and accompanied lighting sources. Finally, the food analysis result is sent to the end user.
2. Deciding, whether to store received measurement data into the training data set or to store data only into the archive - whether the source of data is trusted as an expert user building and validating existing and new data sets.

Once the measurement data and configuration parameters are received through the cloud-mobile API, the push-pull asynchronous messaging mechanism (based on the ZeroMQ library) utilizes internal APIs to store data into the operational database (more details in section 4) and instantiate the rule engine script. This rule engine script is one cloud process thread which listens for analysis requests on an internal API. Once the request is sent, the rule engine pulls corresponding measurement data and measurement configuration parameters from the operational database. Based on the configuration parameters, the rule engine decides, which data analysis and machine learning pipeline to employ for analyzing the measurement data. For now, we have implemented and optimized one data analysis pipeline for each sensor per available data set (combination of project use case and food type). The rule engine mechanism instantiates a new processing thread for the selected data analysis pipeline. Once the data analysis is finished, the rule engine returns analysis results with measurement data identificator to the ZeroMQ process over internal API. The ZeroMQ sends response to the result formatting procedure which maps the measurement data ID with end user ID and sends push notification with the FCM (Firebase Cloud Messaging) service to the corresponding user. Once the

measurement result is sent to the end user it is also pushed to the operational database through the internal API. This way end users can access their measurement history combining measurement data, configuration and analysis results. In the first prototype implementation the analysis result is a vector with three elements showing derived class based on measurement data for NIR and UV-VIS sensors. The vector is organized as [NIR, FLUO, VIS]. Classes have their own labels which are also present in the collected datasets used for the ML model training. Expert users understand these class labels and can easily validate analysis results.

Another process in the reactive decision making framework selects measurement data from the operational database to be included in the training datasets for project use cases. In the first system prototype implementation, this process is performed manually.

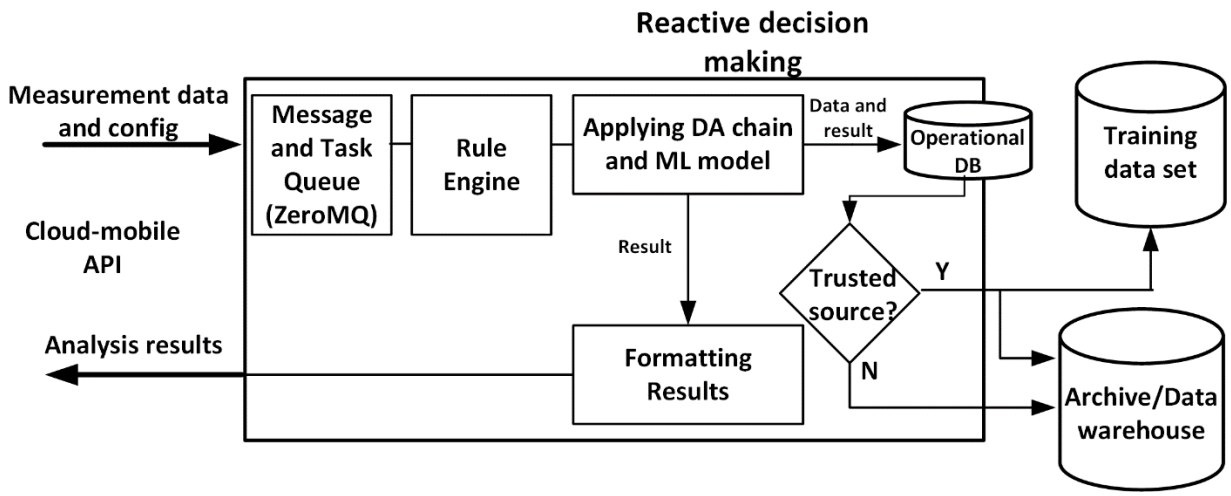

Figure 5 - PhasmaFOOD reactive decision making mechanism

Figure 6 showcases proactive decision making which focuses on strategic and policy based management of the PhasmaFOOD system. Based on predefined schedule (daily, weekly, monthly job) and a set of internal triggers (amount of new data set entries from validated measurements, reached trigger level for the number of errors in classification) as well as changes in regulatory data (collected from 3rd party service), the proactive decision making will employ rule engine directives, which will update the system components to address the detected changes. Output of the decision making will be directives to perform ML model training with the updated data set and to perform calibration of the sensing device and update its firmware software. Another task for the strategic decision making directives is to perform offline analysis of measurements and analysis results provided by untrusted sources. It includes analysis of data from the DB archive, quality assessment and making decision on whether to include measurement into the training data sets. This will ensure that all data that reach the cloud platform are maximally utilized to improve overall performance of the PhasmaFOOD decision making processes.

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

Proactive decision making procedures are executed manually in the first PhasmaFOOD system prototype implementation. We will work on automating processes for the final prototype.

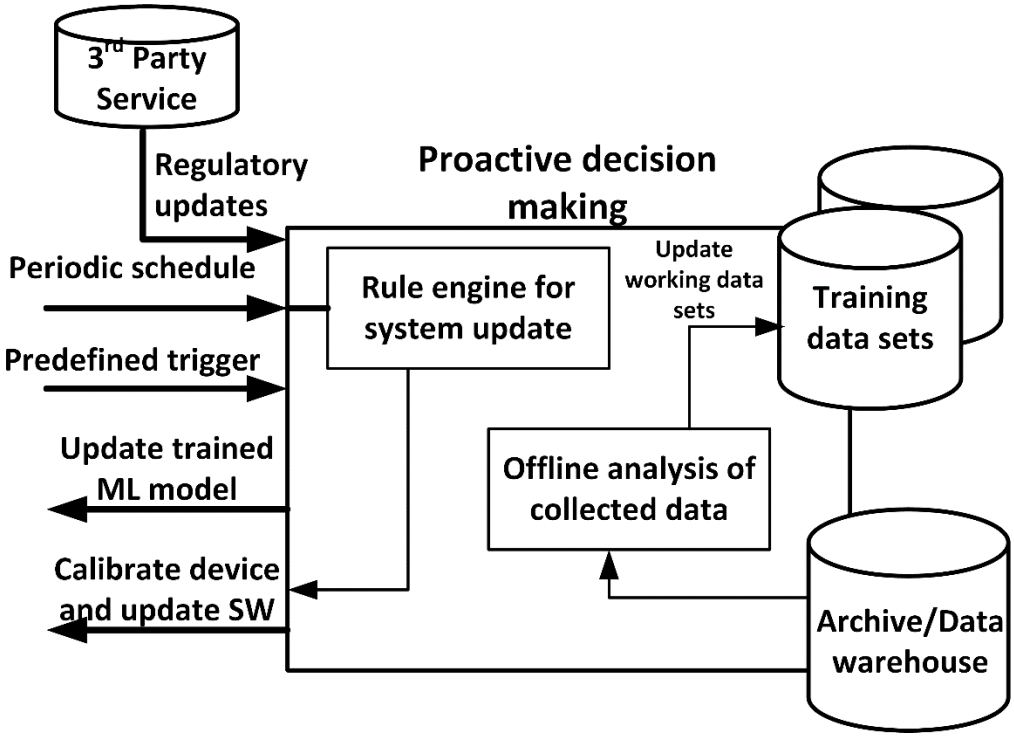

Figure 6 - PhasmaFOOD proactive decision making mechanism

2.3 Matching requirements to data analysis and decision making algorithms

Ref. number	Description	Data analysis and decision making algorithms
DATA-CALIB-3	Data analysis algorithms and related protocols deployed on the cloud platform SHOULD provide calibration recommendations based on specific application.	The decision making procedures which are hosted on the cloud platform, decide on proper data handling and system calibration.
DATA-CALIB-4	Data analysis algorithms and related protocols on the PhasmaFOOD device/smartphone/cloud SHOULD provide 'online' system training/calibration capability to improve on the 'offline' or 'user-initiated' calibration process.	

3 PhasmaFOOD mobile application

3.1 Mobile application architecture

3.1.1 Block-level description

The PhasmaFOOD mobile application (the first prototype) comprises of software modules presented in Figure 7. Each module is described in

Table 1.

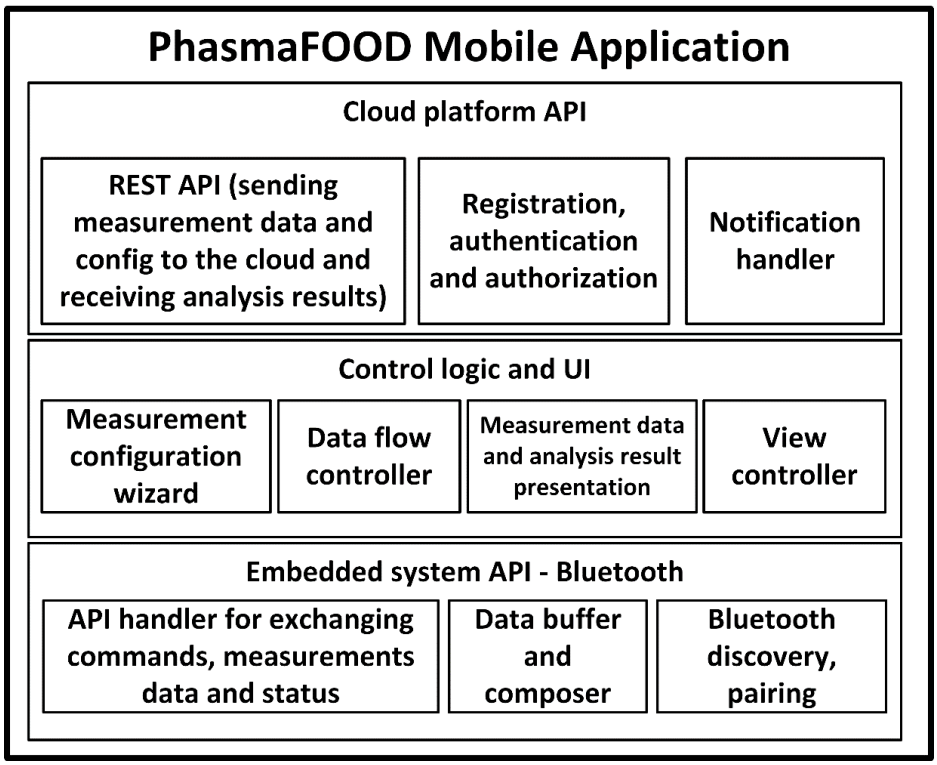

Figure 7 - PhasmaFOOD mobile application block diagram

Table 1 - PhasmaFOOD mobile application building blocks

Module	Sub-module	Functionality
Embedded system API and Bluetooth handlers	API handler	Application side of the embedded-mobile communication API. The handler builds requests from measurement configuration wizard, receives measurement status alerts and preprocessed measurement data from the embedded system.
	Data buffer and formatting	This module is responsible for mapping data and commands exchanged over the embedded-mobile API into and from Bluetooth packets.
	Bluetooth discovery	This module employs Android classes for utilization of

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

	and pairing	the Bluetooth interface and enabling the mobile app to discover the PhasmaFOOD smart sensing device (also utilizing the Bluetooth interface) and establish communication session.
Cloud platform API	REST communication API	Handles cloud-mobile API on the application side. This module maps measurement data, measurement configuration and analysis request into the format specified by the API. The same API handler is used for receiving data analysis results from the cloud platform.
	Authentication and registration	This is also part of the cloud-mobile API with specific focus on exchanging authorization and authentication tokens between the mobile application and the cloud platform. It also enables password recovery. The same mechanism is used for the web dashboard.
	Notification handler	This module manages push notifications from the cloud platform towards the mobile application. For now, the push notifications are used to inform the mobile application that the measurement data are received on the cloud end and when the data analysis is finished and results are ready for presentation. Firebase Cloud Messaging (FCM) service/API is utilized for this functionality.
Control logic and UI	Measurement configuration wizard	Provides UI for measurement configuration by enabling end user to select use case, food type, relevant sample parameters and individual sensor configuration (spectrometers and camera) and then format command for the embedded-mobile API. The wizard is implemented as a tree and directly transforms into a JSON to be included with measurement request.
	Data flow controller	This module is responsible for ensuring data and command flow between all mobile application modules. RxJava is used for creating reactive streams and advanced data processing and execution.
	Data presentation	This module is responsible for presenting analysis results received from the cloud and also presenting in chart form preprocessed measurements received from the embedded system (for quick troubleshooting). Results are presented in a textual form with additional graph for easy VIS and NIR dataset observation and verification. MPAndroidChart is used for plotting charts. This module will be extended with additional features for non-expert users.

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

	View controller	This module is responsible for the UI flow execution in line with the implemented logic. The MVVM (model view view-model) architecture is implemented, which allows efficient layers decoupling, simple testing and seamless app upgrading with new features (see Figure 8). Google Architecture components (Jetpack) are used (LiveData, ViewModel).

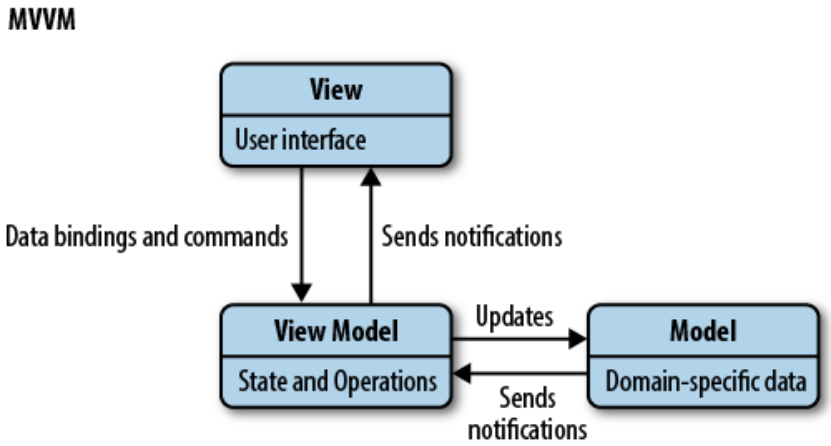

Figure 8 - MVVM typical workflow

The PhasmaFOOD Android application source code repository will be made available on the project website once the first prototype is finalized and validated. The application will also be published on the Google Play store, but it will be kept private (available only to invited testers) until the final prototype is produced. At this stage of the development, the application is distributed among project partners as .apk file.

3.1.2 Usage UML diagram and description

The mobile application usage UML diagram is shown in Figure 9. It is a derivation of the UML usage diagram presented in the D2.1, with focus on functionalities implemented and supported in the first prototype.

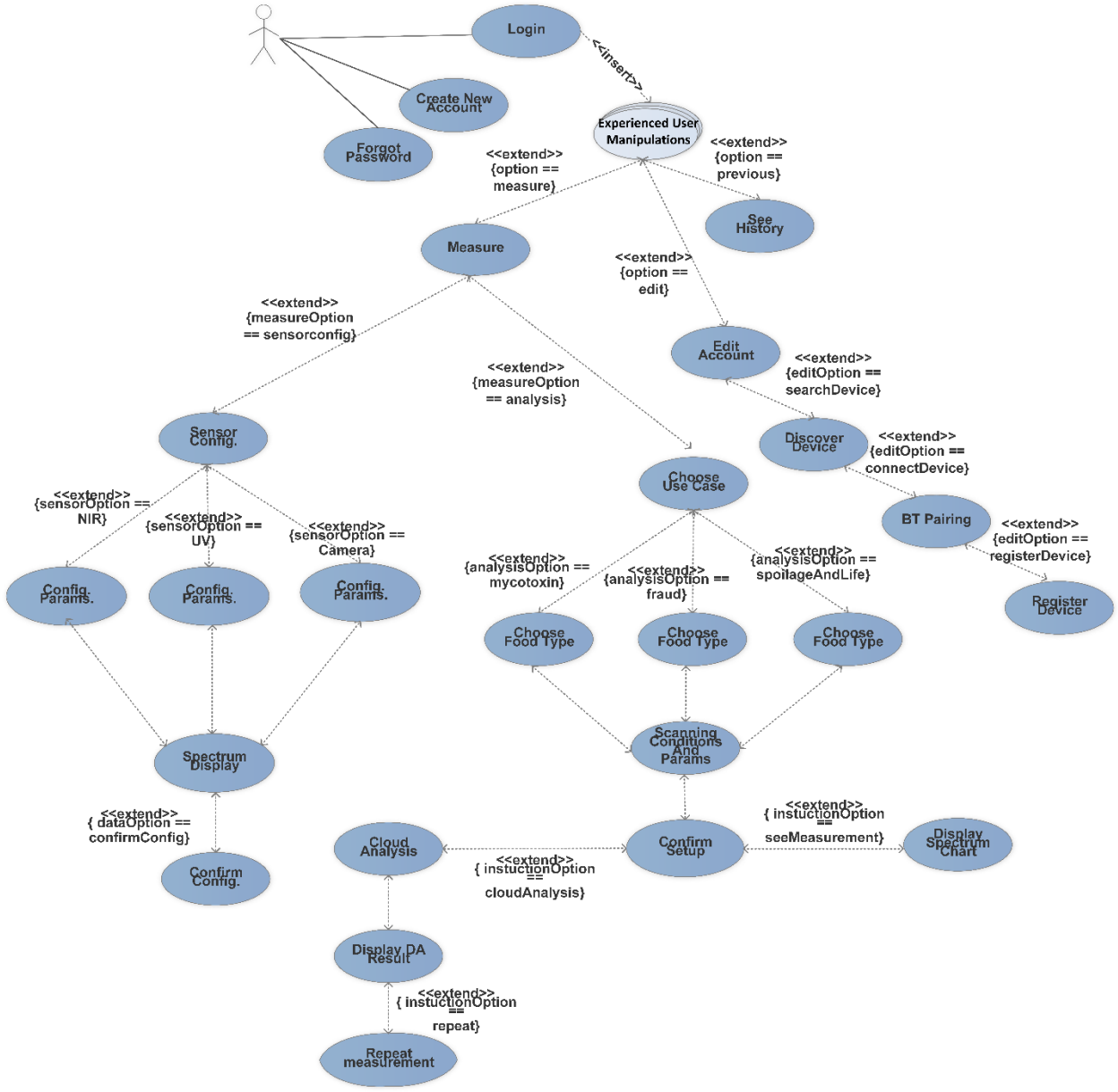

Figure 9 - UML usage diagram of the PhasmaFOOD mobile application

The mobile application flow is implemented so as to properly reflect on experiment protocols used by project partners in laboratories while building and validating use case data sets. The usage flow of the mobile app can be divided into the following view streams:

- User registration and login process - If a user wants to create a new account, this can be done by completing a registration process. Basic data should be provided (first name, last name, company – for project partners, email, username and password). The user is able to reset their password. Also, the user can login after registration.
- Smart sensing device discovery and pairing – The user can scan for nearby Bluetooth devices, choose the PhasmaFOOD smart sensing device and initialize pairing and

connection establishment processes. Once paired, the device is registered under the user's account on the cloud platform. The mechanism will be updated so that only specific Bluetooth IDs are scanned for (the ones belonging to the produced PhasmaFOOD smart sensing devices).

- View measurement history – The user will be able to access previous measurements and analysis results in a specific view.
- Measurement configuration wizard – The end user specifies use case, sample/food type, sample status (relevant parameters depend on the use case and food type) and initializes the measurement process on the smart sensing device.
- Sensor configuration – Once the use case and food type are specified and before starting the measurement process, the user can calibrate configuration parameters of individual sensors embedded into the smart sensing device (NIR, UV-VIS spectrometers and CMOS camera). This will enable expert users to troubleshoot and validate system setups for different food types and experimental conditions.
- Data presentation – Once measurement data (preprocessed) are received from the embedded system, the user will see the collected data for each spectrometer in a form of a chart (sensor specific values on Y axis and wavelengths on X axis). This enables fast troubleshooting. The user will be able to repeat the measurement with the same configuration of the measurement process and with the same or calibrated configuration parameters of individual sensors.
- Result presentation – Once the data analysis result is received from the cloud platform, a separate view is generated in the application showing analysis results with labels used in the training data sets. This view will be different for end users in the final prototype (more informative for non-expert users). Start new data set – This functionality will be implemented in the final prototype. It will enable project partners/use case leaders to start collecting new data sets and specifying new sub use case.

3.1.3 GUI and UX

The user interface and user experience of the first PhasmaFOOD mobile application prototype are designed and specified with expert users in mind. This enables project partners (use case leaders and food experts collecting and analyzing use case data sets) to utilize the PhasmaFOOD system and smart sensing device during collection of new and validation of existing data sets used for training decision making models in the cloud platform.

Figure 10 shows screenshots of the Android application with the Home interface and interface for listing nearby Bluetooth devices, discovering new device and pairing with it. Figure 11 shows screenshots of the measurement configuration wizard with use case 1 – Mycotoxin detection and Almond sub use case. Figure 12 shows the interface after the measurement is configured. Users can configure individual sensors participating in the measurement process (project partners are familiar with working on calibration of provided parameters with legacy software used with NIR and UV-VIS sensors and the CMOS camera). Users can choose to proceed with

default sensor configuration parameters for specified use case and sample combination (previously validated in laboratory). Next, the user chooses to send measurement configuration parameters to the embedded system which employs the sensing unit to collect measurement data, preprocess measurement data and send them to the mobile application. When measurement data are received, the mobile app displays spectroscopy readings in chart form for quick validation and troubleshooting. User can confirm the measurement and send it to the cloud for analysis against specific DA chain. Upon analysis completion, the app receives a notification and pulls results from the cloud platform. For now, results are displayed with labels used in training data sets.

End users of the final prototype will have a subset of options which are offered to expert users. The Android version of the mobile application will be used as specification for building the iOS version for the final prototype.

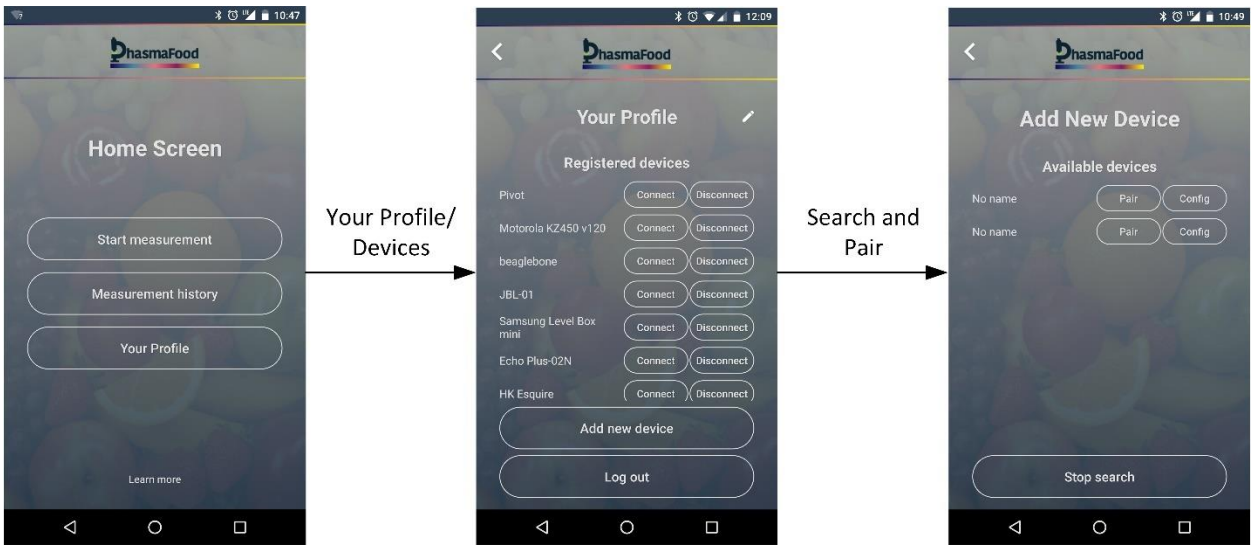

Figure 10 - Mobile application GUI - Hoe screen and device pairing

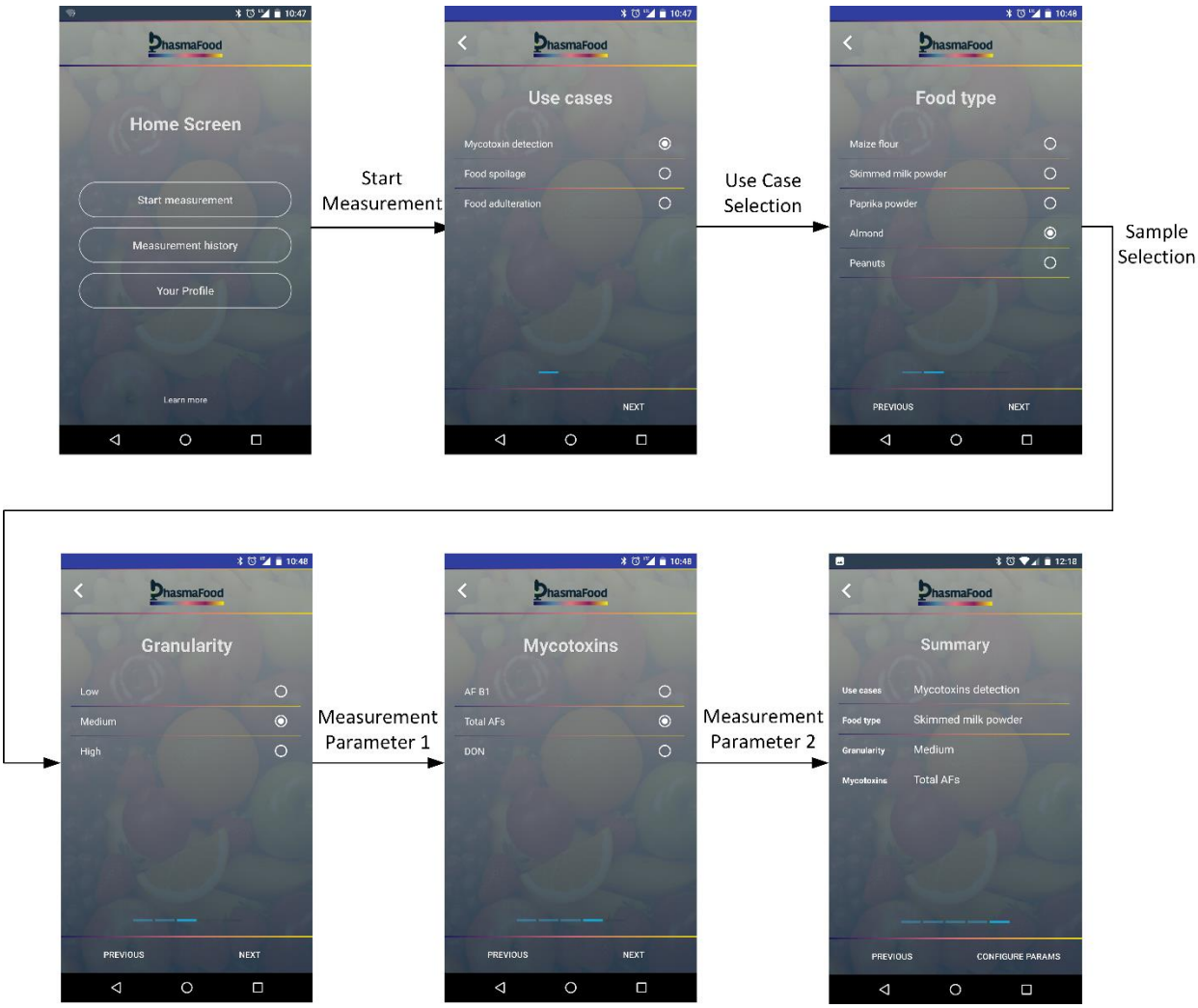

Figure 11 - Mobile application GUI - measurement configuration wizard (Use case 1 – Mycotoxin detection example)

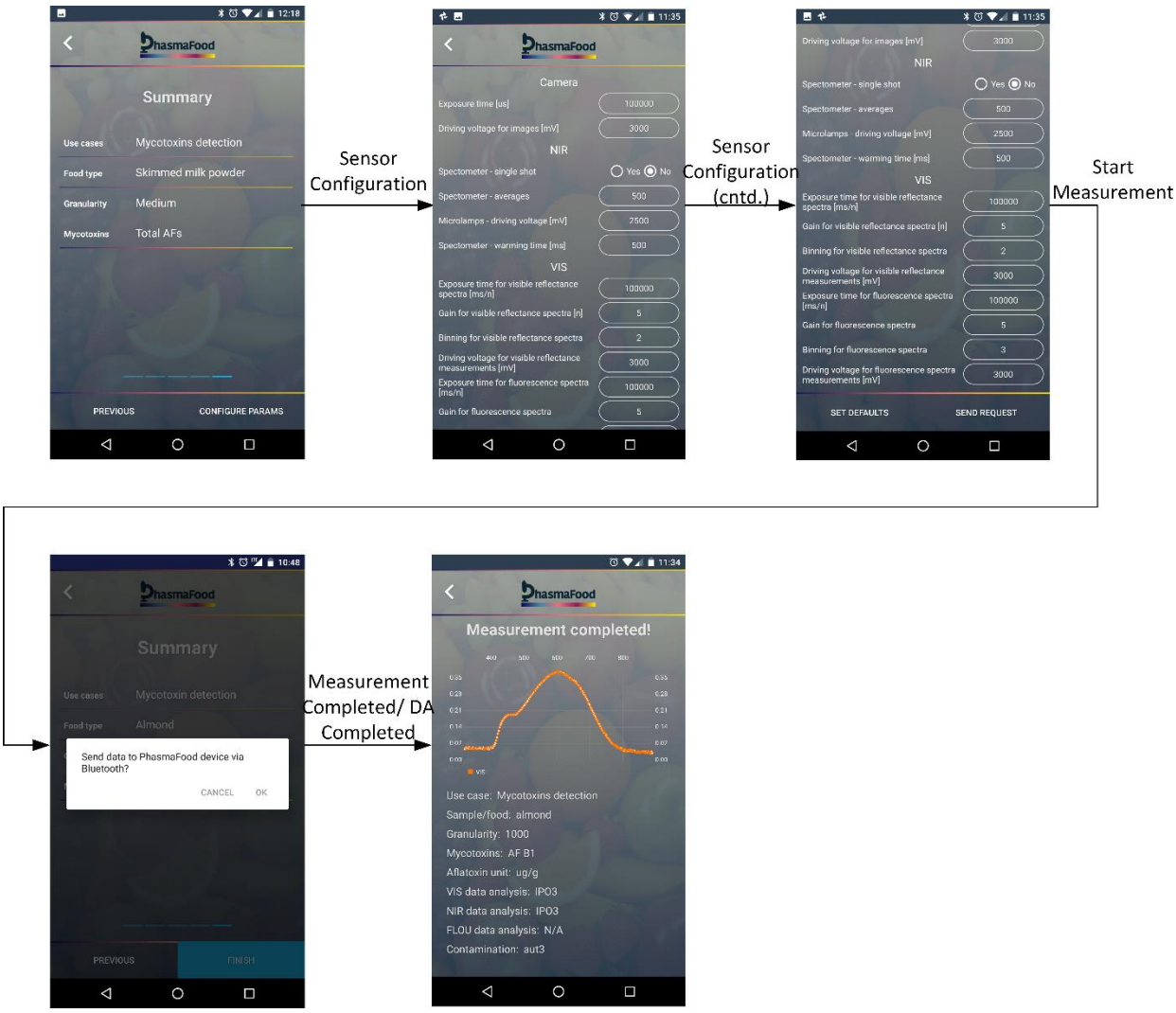

Figure 12 - Mobile application GUI - configuration and individual sensors, measurement data and data analysis result presentation

3.2 Communication with sensing device

3.2.1 Description

As a bridge between the PhasmaFOOD smart sensing device (primary source of data in the PhasmaFOOD system) and the PhasmaFOOD cloud platform (primary destination of all data to be analyzed), the PhasmaFOOD mobile application needs to enable and serve two APIs. These two APIs enable communication path between the smart sensing device and the cloud platform, while at the same time providing the end user with a GUI for configuring the measurement process.

The embedded-mobile API enables the smart sensing device to receive measurement configuration parameters and report on measurement process status and deliver measurement data obtained from the sensing unit and preprocessed with embedded data preprocessing procedures. The API is established over a Bluetooth communication channel between Bluetooth interfaces of the smart sensing device and smartphone hosting the PhasmaFOOD mobile app.

3.2.2 APIs and API handlers

Commands and/or data exchange protocol/sequence, abstract design, pseudocode.

The embedded-mobile API sequence UML diagram is shown in Figure 13. This sequence diagram shown the main phases in establishing the communication channel and performing command and data exchange between the embedded system and the PhasmaFOOD mobile application. All presented sequences are implemented in the first PhasmaFOOD system prototype.

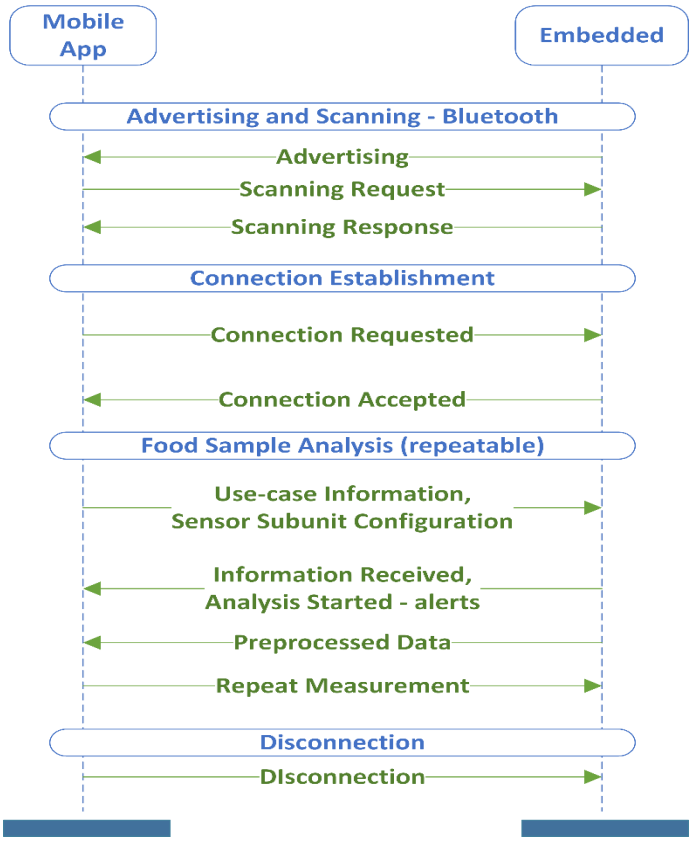

Figure 13 - UML sequence diagram of the embedded-mobile API

The embedded-mobile API includes the following phases:

- Bluetooth discovery and pairing:
  - In this phase standard Bluetooth protocol procedures are used for enabling the mobile application to discover the PhasmaFOOD smart sensing device

(advertising mode on both ends). Next, pairing procedures are implemented for establishing the communication channel.

- For discovering, standard Android Bluetooth library is used. It is wrapped with an additional layer that creates reactive streams (RxBluetooth) for sending data, receiving data and observing Bluetooth states (discovering, bonding, connection etc.).

- Establishing Bluetooth communication channel
  - In this phase, the Bluetooth communication channel is established between the embedded system and the mobile application. The Bluetooth software driver on the embedded system partitions the measurement data into datagrams for the Bluetooth protocol packets. On the mobile application side, a software module is employed for buffering and combining received Bluetooth packets (can be sent out of sequence) and correctly assembling the measurement data to be further transmitted to the cloud platform.
- Measurement configuration and reporting
  - Data and commands exchanged between the mobile device and the embedded system are in JSON format. The measurement data are sent in the same JSON data format, which is used in the cloud DB (described in the section 4).
  - The following is a typical operational sequence of this API phase:
    1. API handler of the mobile application sends measurement configuration command in JSON format. This message specifies use case, food type, sample conditions and configuration parameters of individual sensors (spectrometers, light sources and camera). If sensor configuration parameters are not specified, default values are sent to the embedded system. These default values are defined by use case leaders while collecting relevant data sets.
    2. The embedded system acknowledges reception of the configuration parameters and starts the measurement process. During the measurement process the embedded system can send alert messages to the mobile application in case there are problems with the measurement process. The user is informed to make appropriate corrective action and retry the measurement. Alerts and embedded system triggers are reported in D5.3 and D2.1 in description of embedded system state machine. Corresponding end user's corrective action will be decided through validation of the integrated first system prototype.
    3. Once the measurement is finished the embedded system formats data into the JSON format and sends it over the Bluetooth channel. The mobile application acknowledges reception of every packet and the complete sequence of the data stream.

4. Measurement data is presented to the end user in chart form and the user can choose to send it to the cloud platform for analysis, send them to the cloud platform for archiving (for reporting purposes – not to be analyzed) or discard them to save cloud storage and communication load. The mobile application further formats the received measurements and appends measurement configuration parameters to the JSON message and sends it to the cloud platform. The user can choose to repeat the measurement with the same setting. In this case the API handler sends a repeat command and the embedded system knows to repeat the measurement process with the last configuration parameters.

- Disconnection of the Bluetooth channel between the embedded system and the mobile application

Different modes are investigated and validated. The API logic enables the embedded system to initiate the disconnection procedure upon successful reception of the measurement data stream by the mobile application. In another mode, the communication channel is disconnected after 2 minutes of inactivity. Currently, this time is considered indicative and may be adjusted during trials. The channel can also be disconnected on the smartphone running the PhasmaFOOD mobile application.

3.3 Communication with cloud platform

3.3.1 Description

The cloud-mobile API enables the mobile application to register the user on the system and establish a secure communication channel by using JWT (JSON Web Token) tokens. Then, RESTful procedures are used to send measurement data and configuration parameters to the cloud platform. A FCM notification is sent once the measurement data are analyzed and the mobile application pulls results from the cloud platform in JSON format. This API can be established over any Internet connection (WiFi, 3G, LTE, USB to Ethernet).

3.3.2 APIs and API handlers

Commands and/or data exchange protocol/sequence, abstract design, pseudocode.

Figure 14 shows the UML sequence diagram of the cloud-mobile API. All data exchange is authorized with JWT tokens, exchanged between the cloud platform and the mobile application. There are three phases in API utilization:

- Registering a new user and authorizing a mobile application instance to exchange information with the cloud platform. Successful registration creates a user profile in the cloud DB and an authorization function to be used for verification of API tokens, sent by the mobile application for each new API message. Tokens are refreshed on each new login and after a defined period of time. Each registered user will have cloud profile and will e able to save measurements, analysis history and configuration parameters in the

operational database. In addition to that, expert users will have ability to save complete new data sets, update existing datasets (with permissions) and train new ML models.

- Profile management phase of the API enables mobile application users to register a new smart sensing device under their profile. Also, users can update the login password. Finally, users can send a request to get their measurement history, showing all previous measurement results with measurement configuration and timestamp.
- In the food sample analysis phase, the RESTful API procedures are utilized to send measurement analysis requests, accompanied with measurement data and measurement configuration parameters in JSON format. Based on the measurement configuration parameters, the cloud platform decides, which data analysis chain to utilize on the received measurement data. The mobile application is subscribed to the FCM push notification system upon registration. Once the data analysis is finished, a push notification is sent to the mobile app and it then pulls results from the cloud platform (API request).
- Since the REST API does not establish a session between mobile app and the cloud platform, there is no need for disconnecting the communication channel.

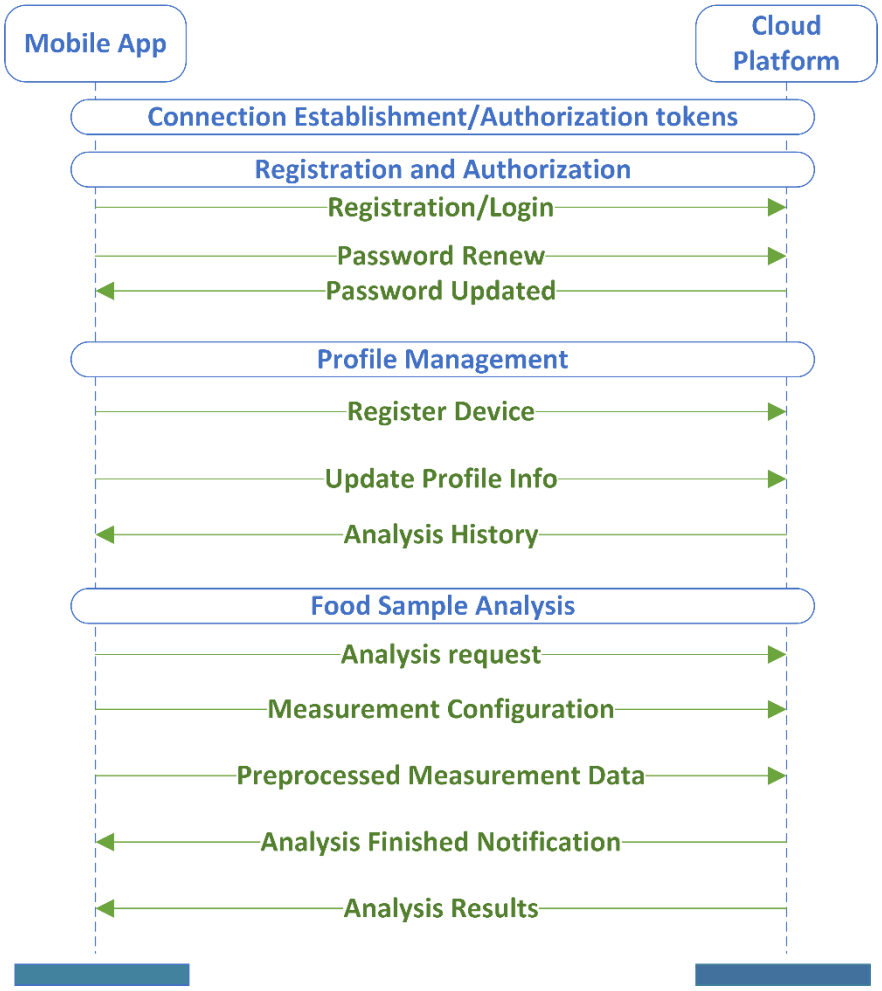

Figure 14 - UML sequence diagram of the cloud-mobile API

This API is the basis for building the publicly available API for 3rd party developers and integrators adopting the PhasmaFOOD solution. The interactive API documentation is under preparation.

3.4 Matching requirements to mobile application development

Matching of related data communication requirements are reported in D5.3 Section 4.

4 PhasmaFOOD cloud platform

4.1 Cloud platform architecture

The PhasmaFOOD cloud platform is the destination for all collected measurement data and datasets used for training decision making models. The cloud platform hosts the PhasmaFOOD

cloud database with all use case datasets and provides services for the mobile application and the embedded system. Key functionalities of the PhasmaFOOD cloud platform are:

- To host the PhasmaFOOD cloud database and data warehouse for storing use case data sets as well as user profiles, trained ML models, decision making history and regulatory parameters.
- To perform data analysis, using machine learning pipelines on provided measurements, specific to each project use case and sample/food type.
- To execute decision making procedures and rule engines and ensure proper data flow throughout the PhasmaFOOD software system.
- To authenticate and authorize end users.
- To provide a web dashboard for end users and system administrators for viewing and managing data sets and measurement history, training new ML models and building new data sets.
- To host the handler for the cloud-mobile API, described in section 3.To host 3rd party API to be opened for developers and PhasmaFOOD system integrators.

4.1.1 Block-level description

Block diagram of the PhasmaFOOD cloud platform is presented in Figure 15. The main building blocks of the cloud platform are listed and explained in Table 2.

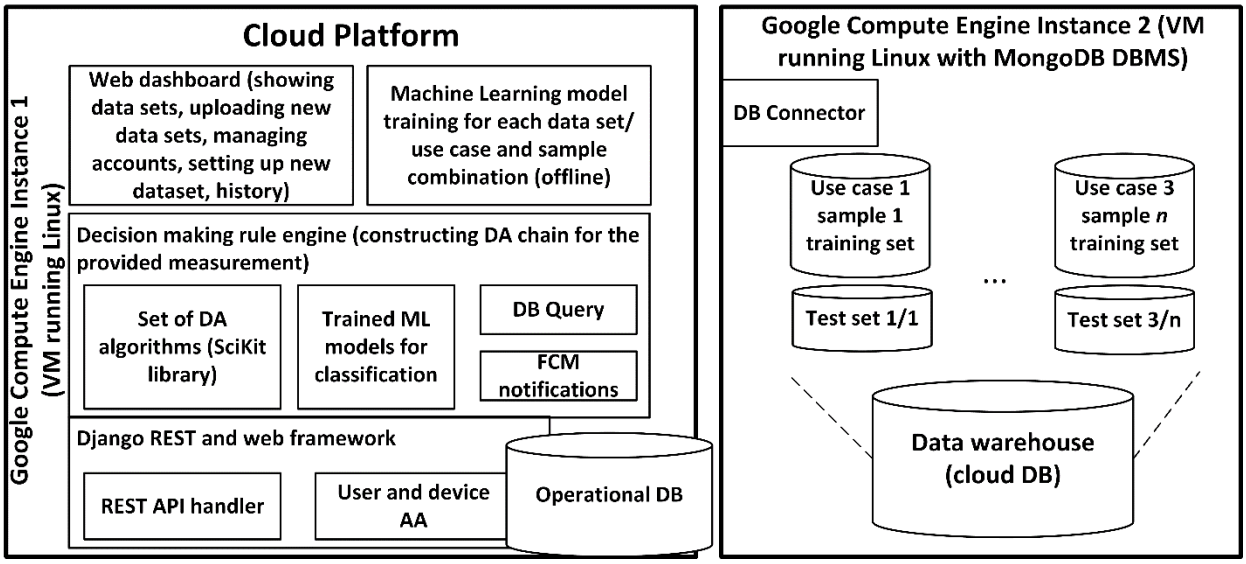

Figure 15 - Block diagram of the PhasmaFOOD cloud platform

Table 2 - The main building blocks of the PhasmaFOOD cloud platform

Module	Sub-module	Functionality
Django REST and web	Cloud-mobile REST API handler	This module hosts the cloud-mobile API handler as Django REST framework. It enables exchange of measurement data, configuration parameters and

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

framework		analysis results between the mobile application and the cloud platform. API details are provided in section 3.
	Web dashboard API	This API is extended version of the mobile-cloud API and it enables exchange of data, parameters and commands between PhasmaFOOD web dashboard and the cloud platform backend processes. Django REST framework is utilized.
	User registration and authorization	This module and API is responsible for enabling user registration and authentication for authorized data exchange and resource/service access by end users. The JWT token mechanism is used. More details in section 5.
	Data flow controller	This module is implemented as asynchronous messaging framework by using the ZeroMQ library and internal APIs implemented in Django framework. The ZeroMQ publish-subscribe (or push-pull) mechanism is used for enabling creation of tasks or instantiation of the decision making processes described in the section 2. The basic ZeroMQ operation is depicted in Figure 17. The Django web framework is configured as publisher – server. Backend decision making processes are configured as ZeroMQ subscribers who listen to subscribed topics published by the Django server. This module also includes internal REST APIs for exchanging data between the decision making processes, end user APIs and the cloud databases.
Decision making rule engine	DA and ML pipelines	These pipelines are implemented using the Python Scikit-learn library. These pipelines comprise steps, which implement data analysis/preparation algorithms. Final step in a DA pipeline is trained ML model for classification or regression. Each use case – sample-sensor combination has its own ML model. The pipeline scripts are executed in rule engine threads. Analysis results are sent to the end user and stored in the cloud DB.
	Rule engine process	This module is responsible for instantiating new data processing threads and running data analysis pipelines

		based on received measurement configuration parameters. The rule engine process is described in section 2 under reactive decision making.
	DB query	DB query module is the backend process, which establishes API with the cloud DB and enables exchanging data between the decision making processes, mobile app, web dashboard and the data set database as well as operational database.
	Push notifications	This software module employs Firebase Cloud Messaging (FCM) service for pushing notifications from the cloud platform to the PhasmaFOOD mobile application. For the first system prototype we use FCM notifications to inform the mobile application that the data analysis process for delivered measurement data is finished and can be sent to the end user. Example of a typical FCM exchange sequence is shown in Figure 16. In this scenario, the PhasmaFOOD cloud platform is the Server app and PhasmaFOOD mobile application is a Client app. Message of the push notification contains food analysis result (value vector described in the section 2.2.).
Operational database	This is PostgreSQL database utilized by the Django framework for storing user submitted data and actions. It stores user profiles, measurement history, data analysis results and data on registered PhasmaFOOD sensing devices.	
Web dashboard	GUI	The UI of the web dashboard is implemented in Angular 5 framework. It supports sequences shown in Figure 19.
	API handler	This API handler enables data and instruction exchange between the web dashboard frontend and the PhasmaFOOD cloud platform backend processes and the cloud database. It is implemented with the Django REST framework. It follows the logic of the cloud-mobile API described in section 3, but also includes additional directives for: • Enabling administrator level services (overall monitoring of the service and database contents).

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

		• Enabling cloud DB queries for listing datasets. • Enabling end users to configure new datasets to be collected with the prototype solution.
	Web server	For serving the web dashboard frontend and UI we have deployed and configured the NGINX (Engine X) web server.
ML model training	These proactive decision making processes are currently executed manually from the cloud platform command line. It comprises Python Scikit-learn library scripts and cloud database (dataset) connector. The first prototype will enable execution of the ML training process from the UI of the web dashboard.	
Cloud Database	DB connector	Provides API for querying the cloud DB and getting dataset entries.
	Document parser	This module comprises document parsing scripts used for transferring data from documents (tables produced by laboratories using legacy dataset collection solutions) into the data model format used in the cloud DB.
	Dataset splitting	This module produces training and testing portions of the use case datasets stored in the cloud DB. This splitting is done randomly to ensure training of non-biased models.
	DA pipelines	This is a collection in the cloud DB, which stores configuration parameters for Scikit-learn library DA pipelines.
	Data warehouse	This is a MongoDB database with document collections comprising PhasmaFOOD use case datasets in specified data format.
	Data archive	This is a MongoDB document collection, which will store periodic backups of the data warehouse and operational database.

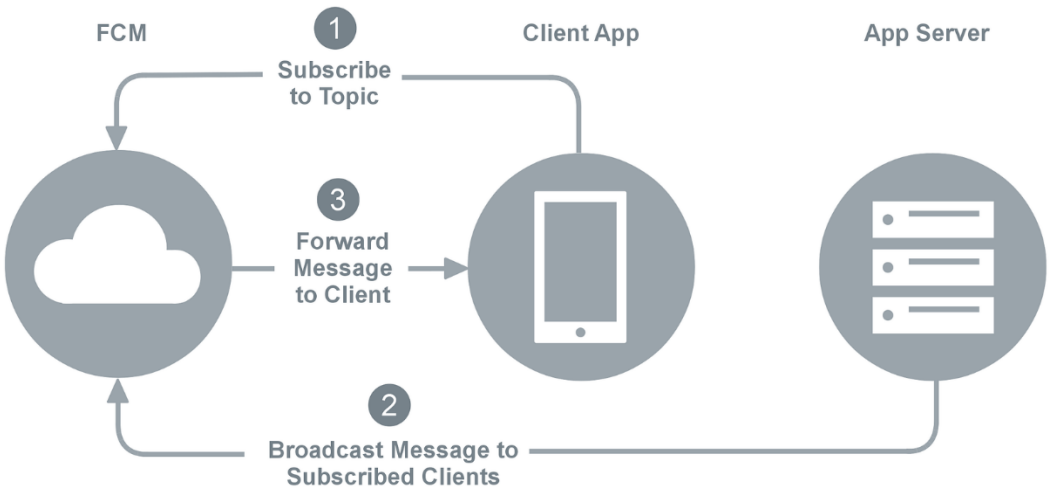

Figure 16 - Firebase Cloud Messaging mechanism

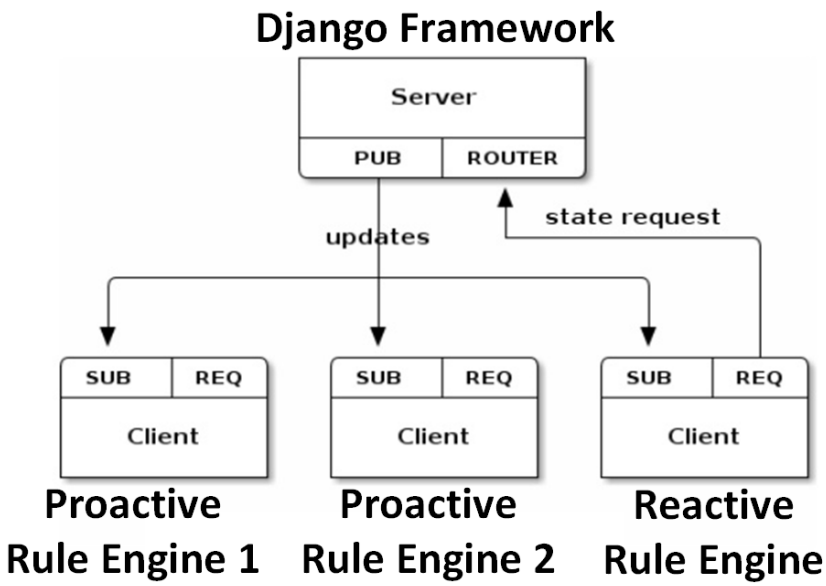

Figure 17 - Basic operation of the ZeroMQ publish-subscribe mechanism

4.1.2 Data model for collecting laboratory data

The PhasmaFOOD cloud platform facilitates the analysis and use of laboratory spectroscopy data gathered by the experiments conducted in related WP3 tasks. The experiments involve several replicates and provide data for all the identified project use cases, incorporating a wide

variety of food types and levels of sample contamination. The information provided by the experiments data is crucial to the derivation of test and training data sets, the successful ML model training and validation tasks. It is necessary that easy experimentation and access to available information is supported. To enable adequate experiment data storage, usage and manipulation that serves the above needs, a specific internal structuring of the data was adopted, when persisting information to the cloud database. This structure can be seen in the corresponding appendix (Appendix A) at the end of this document, which shows a reference JSON Schema [8].

In the course of an experiment, a specific laboratory will consider several samples related to a particular use case and spectroscopy data are gathered from multiple sensors. Furthermore, the experiments are organised in replicates that contain multiple samples per replicate and may span several days. The information described has dedicated data model fields to accommodate the storing of sampleID, laboratory, foodType, useCase, replicate, and dateTime. Among others, the data model also introduces fields for storing information with respect to the temperature and its variation patterns and the specific exposure hours where a sample has remained at a particular temperature. The corresponding fields are temperature, temperatureProfile and tempExposureHours. In order to capture the packaging information of a particular sample, a dedicated field is provided (packageType) and also there is a specific field for storing the granularity of a sample wherever applicable (granularity).

With respect to the information related to the contamination of a sample, it has been identified that the Total Viable Count of microorganisms is of course necessary to be stored and at various cases, there exists a contamination label that the laboratory can provide about the sample used. Furthermore, the Aflatoxin Quantification is also maintained. The relative fields that hold microbiological information are microbiologicalUnit, which maintains the unit that may vary across samples and the microbiologicalValue. Specifically, for storing aflatoxins related information the name, quantification and applicable unit are stored, whenever necessary. Similarly, the data model can accommodate adulteration information in dedicated fields (adul).

While the parts of the data model described so far maintain significant information about the sample in an experiment, the actual spectroscopy data from various sensors are of equal importance. To efficiently store the data provided from various sensors during a laboratory experiment, the data model provides the necessary organization according to the sensor type. Fluorescence measurement data are maintained under FLUO whereas visible spectrum measurements are maintained under VIS. The NIR part of the data model is for the near infrared measurements and related images of the sample can be maintained at MSI_IMG. However, it is worth mentioning that the un-processed images from future training datasets will not be stored directly in the mongoDB, since they exceed the 16Mb storage limit. It should be also noted that several measurement files are produced per sensor when measuring a sample. In addition to the measurement files, Dark and White reference files are also produced. Since the data sets must be split for subsequent training of the models, there exists a testData

and a trainData part where the measurements and their corresponding reference values are split according to a selected ratio. The related Dark and White reference values are stored when applicable in dark and ref arrays respectively. The applicable wavelengths (spectrum) of a specific sensor are stored once per sensor type in wavelengths. The data model also accommodates for the storage of averageData values taken across measurements and reference files and can also accommodate preprocessedValues.

4.1.3 Structure of the cloud database

The PhasmaFOOD cloud platform hosts the PhasmaFOOD data warehouse, which stores all collected sensory measurements and contextual data coming from sensing devices, mobile application and 3rd party services like regulatory databases. From this main database, the PhasmaFOOD cloud platform procedures derive data sets used for training ML models for decision making required by each project use case and scenario.

The Figure 18 shows the high-level structure of the PhasmaFOOD cloud database. The database comprises the following entities:

- Operational DB – maintains necessary information and settings for the PhasmaFOOD platform to operate, including user data, regulatory data, information about registered devices and their settings, ongoing measurements and classifications and analysis results.
- Users - stores user related data.
- Regulatory data - contains information from regulatory bodies guiding decision making processes and measurement classification.
- Sensing Devices - stores data related to PhasmaFOOD devices.
- Measurements and results – maintains measurements taken from users of the system, settings related to the device used and its configuration and analysis results while allowing the definition of new data sets.
- Data warehouse - the focal point for all measurement and contextual data. It stores all collected measurements and analysis results. From here the cloud platform procedures build use case specific training data sets, identify data fusion opportunities and provide data for offline analysis guiding proactive decision making.
- Archive - stores analysis results and history of conducted measurements. It is used for offline data analysis and safe keeping the data warehouse state in case of data corruption.
- Training and test data sets - derived from the data warehouse for each use case and scenario that require specific data fusion and ML model training procedure. It is the working database for training and testing ML models and making reactive decisions based on the current measurement provided by the end user.

- Stored DA Chains and ML models - this database stores necessary references to stored Data analysis chains and trained ML models so that they can be easily employed by SW behind decision making processes.

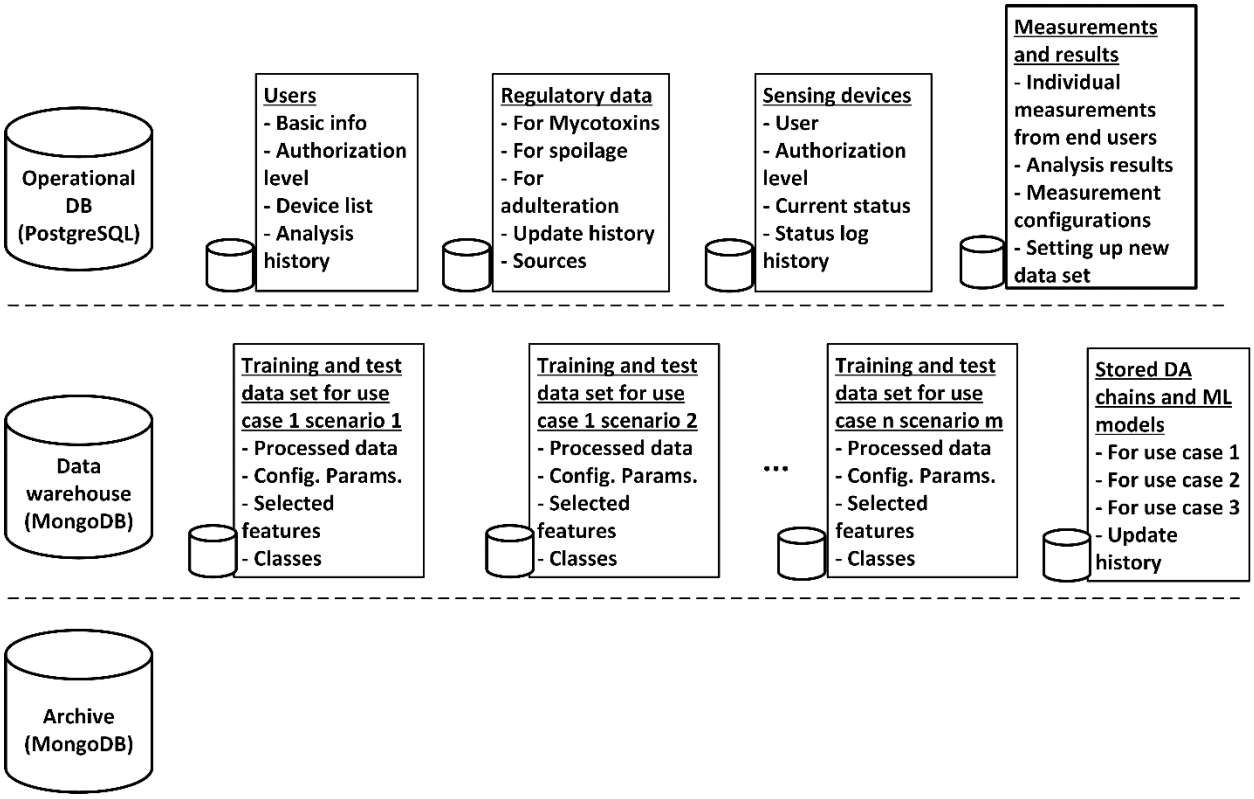

Figure 18 - PhasmaFOOD cloud database

The PhasmaFOOD cloud platform provides reactive and proactive decision making based on collected data sets, provided measurements from end users and system defined triggers. These algorithms are the basis for realization of project use cases, system management and strategic planning of monitored supply and production chains. Decision making mechanisms are implemented as collections of software procedures and algorithms, which take as inputs trained ML models, the current measurement/trigger on which to conduct analysis and employ rule engine to execute all the steps leading to the final decision whether it is reactive decision or strategic/proactive decision. Specification of the proactive and reactive decision making and accompanying block diagrams can be found in the D2.1 [1].

4.1.4 Hosting platform description

For the hosting platform we have chosen the Google Cloud Platform. Currently, the PhasmaFOOD cloud platform is deployed on two Google Compute Engine instances/virtual machines (as shown in Figure 15):

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

- Instance 1 – Cloud services, web dashboard and API handlers
  - Instance type: n1-standard-1 (1 vCPU, 3.75 GB memory) - Intel Broadwell, 10GB SSD
  - Operating system: Ubuntu 16.04.4 Xenial
- Instance 2 – Cloud database
  - Instance type: n1-standard-1 (1 vCPU, 3.75 GB memory) - Intel Broadwell, 10GB SSD
  - Operating system: Ubuntu 16.04.4 Xenial

Docker container images are in preparation for both cloud platform instances, which will enable portability of the PhasmaFOOD cloud platform instances on other hosting platforms.

4.1.5 Dashboard description

The PhasmaFOOD cloud platform hosts the web dashboard. Web dashboard development is work in progress and will be reported and described in detail in the forthcoming deliverables D6.1 and D4.2. The web dashboard is available on this URL: *http://dashboard.phasmafood.eu/*

The web dashboard will be used by system administrators, end users and stakeholders who operate the PhasmaFOOD solutions. The web dashboard is developed in two main phases. The first phase provides a dashboard, which will guide experimenters and project partners during implementation and validation of PhasmaFOOD project use cases. This is the focus of web dashboard development in the first solution prototype. In the second phase, we will develop web dashboard functionalities for non-expert end users. The UI will be simplified, and it will offer a subset of functionalities provided for the expert users (project partners). Figure 19 shows UML diagram of web dashboard usage.

The dashboard recognizes three main classes of users. Expert users/researchers (project partners and food scientists), non-expert users and system administrators (again project partners). The Expert user branch of web dashboard is currently being developed. It will also combine some of the system administrator functionalities. Focus is on enabling project partners to properly configure, calibrate and validate datasets, measurements and analysis procedures produced by the project use cases. The current implementation of the PhasmaFOOD web dashboard enable end users to:

- Register on the platform and login to their profile.
- See PhasmaFOOD application instances and registered sensing devices under their profile.
- See and manage history of obtained measurements and analysis results. Measurement data will be shown in chart form (wavelength on X axis and value on Y axis) with option to toggle table view. Also, measurement will be shown in more user friendly manner with summary statistics.

Expert users will have the same functionalities as end users and in addition they will be able to:

- Try different analysis pipelines and ML models on collected data sets.
- See measurement data details, charts and complete data entries (of their measurements).
- Configure collection of a new dataset under existing project use cases, i.e. user will be able to start collecting data set for a new sample type or sample state under one of the three project use cases. This will ensure that the system integrators will be able to specify new business opportunities with the PhasmaFOOD system.

Administrator users (project partners) will be able to do all that expert users can do, but also:

- List all mobile application instances, all registered users and all registered sensing devices.
- Query all datasets and filter them by use case, sample type and sample status.
- Combine new data analysis pipelines with provided algorithms.
- Select measurement data from the operational DB to be pushed to the cloud DB storing the training data sets.
- Trigger training of new ML models on the same dataset.

The web dashboard backend is served by the NGINX web server while the frontend is implemented in Angular 5 framework and standard libraries for UI elements. REST API between the web dashboard and the cloud services and DB is provided by the Django REST framework.

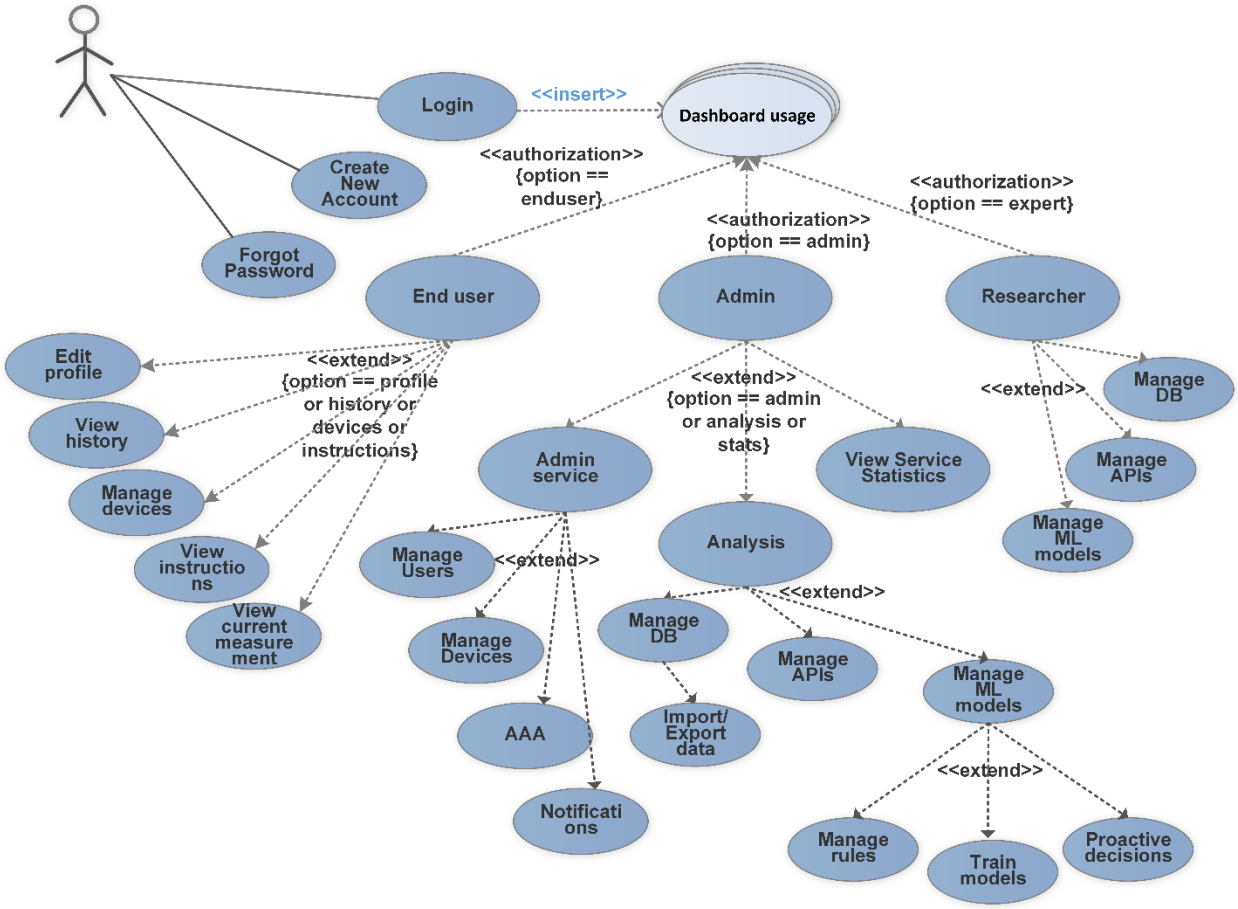

Figure 19 - UML usage diagram of the PhasmaFOOD web dashboard

The next two figures present screenshots of the current implementation of the PhasmaFOOD web dashboard. The UX is still in development, but the main communication and presentation channels for Admin level users are implemented and tested. Figure 20 shows the landing page with Log-In interface. Figure 21 provides first version of the web dashboard interface for presenting measurement history data with charts for selected measurement entry. On the left side there is a menu for shifting between the main dashboard view of datasets, measurement history and machine learning model training. Also, there are options for Admin users to access and manage all registered PhasmaFOOD sensing devices, devices registered under user's profile and list of all registered system users.

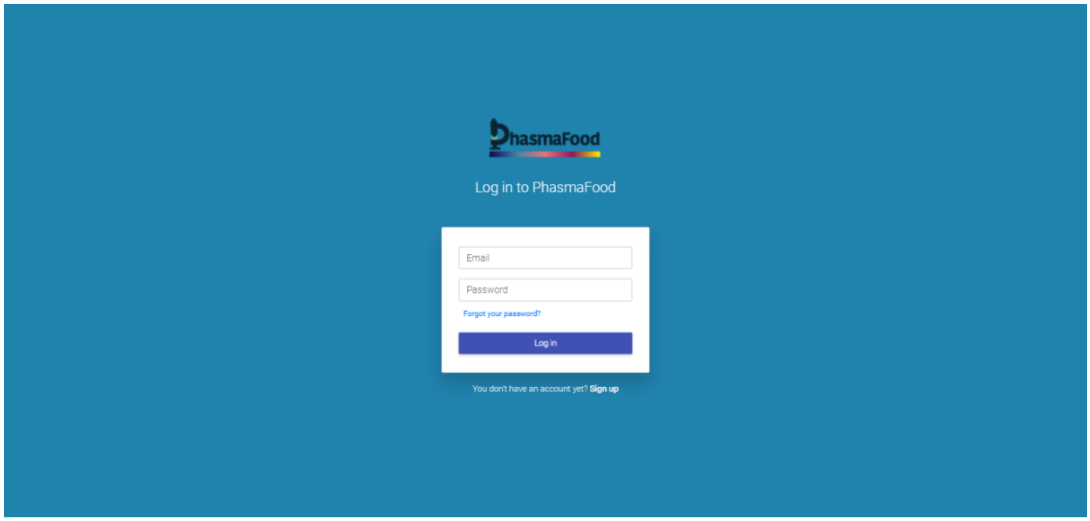

Figure 20 - Web dashboard - landing page (login)

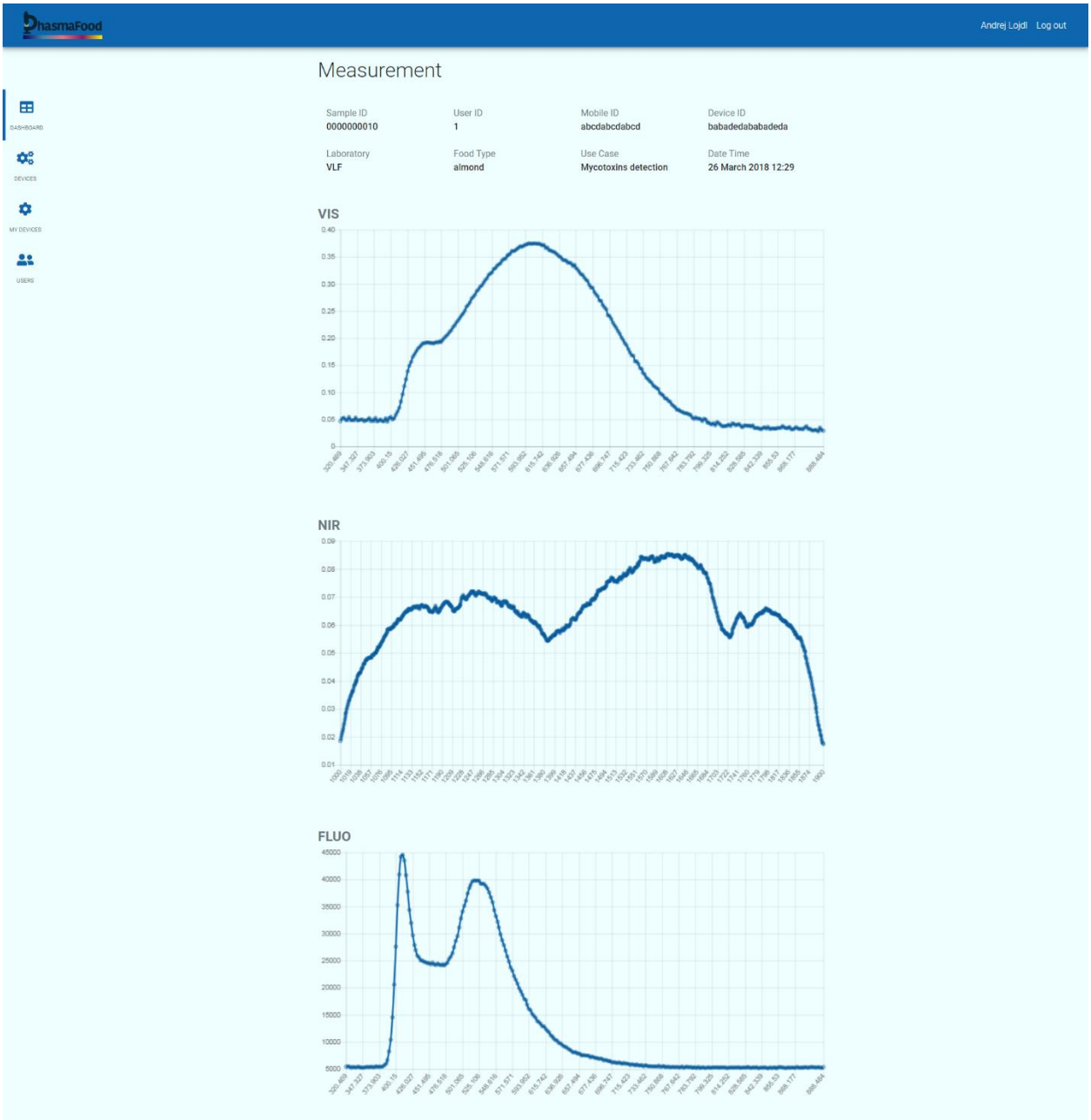

Figure 21 - Web dashboard - preliminary approach for presenting individual measurement data

4.2 Cloud data analysis and machine learning models

The PhasmaFOOD cloud platform hosts the machine learning framework for the analysis of already collected data and newly incoming measurements. It provides models for each use case, food type and sensor combination. Due to its versatility and the availability of many high quality open-source packages for database access and machine learning, the machine learning

framework is implemented in Python 3 [9]. The machine learning framework contains methods for loading training data, training new models, storing the trained models and deploying them for use in the PhasmaFOOD software architecture. The detailed procedures are described in the following sections.

4.2.1 Access to the training database

All collected datasets are stored in the PhasmaFOOD cloud database as documents in a MongoDB. Each datum may contain measurements from the NIR spectrometer, the VIS spectrometer in reflectance and fluorescence mode and possibly from the camera. It contains ground truth values for use case specific target quantities of interest, e.g. for use case 1 aflatoxin counts and/or class information indicating the level of contamination. These ground truth values were obtained in laboratory measurements by food experts and enable the application of supervised learning techniques to train accurate machine learning models. The training database is split into train- and test sets for each collected food type. This practice allows the machine learning models to be trained and evaluated on different data, and hence to obtain unbiased performance estimates. The connection to the MongoDB is established using the *MongoClient* class from the package *pymongo* [10]. For convenience, a function *connect_to_db()* is implemented, in which the host and port as well as a user name and password for the MongoDB are used to connect to the cloud database and return a *client*. The train and test data is queried via a uniform interface *load_dataset* that takes the following input parameters:

- A database *client* as described above,
- The *usecase* and *foodtype* as Strings,
- A list of *sensors*, from which data is to be returned
- A list of *targets*, where each target is a String with the field name that holds the values that will be predicted by the machine learning algorithm
- A dictionary with additional constraints (i.e. here data from one specific date or measurement session can be obtained by specifying these constraints)
- Booleans *traindata* and *testdata* that indicate, whether the train- or the test data shall be returned

A MongoDB query is formed out of this information and the data is returned in two arrays, *X,* of shape *(n_samples, n_features)* and *Y* of shape *(n_samples, n_targets)*, holding the sensor data and the target values, respectively. Internally, for each *usecase* and *sensor* combination a separate query function is employed due to the heterogeneous structure of the so-far acquired training data sets. The sensor data is already reference corrected (when dark and white references are present in the dataset) and in the correct format to be fed into a machine learning model.

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

4.2.2 Defining machine learning models

For the definition of machine learning models, the open-source library *scikit-learn* [11] is used. It contains a wide variety of commonly used machine learning methods, pre-processing steps and evaluation metrics, and is easily customizable. Different processing steps can be chained together using the *sklearn.pipeline.Pipeline* object. A typical machine learning pipeline may contain the following steps:

- Pre-processing (normalization, scaling, discarding low-variance features)
- Feature engineering (PCA, polynomial features, …)
- Feature selection (univariate selection or model based)
- A classifier/regressor for the final prediction

The *Pipeline* object takes a list of 2-tuples, each tuple consisting of a String with a unique name for the step and a *Transformer* object realizing the processing step. The final step of the pipeline should implement a *predict* function, whereas all the other steps only need to implement *fit* and *transform*. While the parameters for each step can be specified directly at the setup stage of the pipeline, it is advisable to tune some hyper-parameters specifically to the dataset at hand. This is usually done via a grid search over a range of pre-specified values and k-times cross-validation to find the optimal parameters. In *scikit-learn* this procedure is realized in the *sklearn.model_selection.GridSearchCV* object. It takes a model (e.g. defined as a *Pipeline* as above), the parameter grid and some technical parameters (the number of folds for cross-validation, the number of kernels to run the process on, etc.) and can then be used like any other *Estimator*, by calling its *fit* and *predict methods.*

4.2.3 Model training and storing

The model training itself then consists only in calling the model's *fit* method, passing it the *X* and *Y* arrays holding the training data and the target variables.

The trained model is then stored on disk in a binary format. This is done with the python package *pickle* and realized in the function *store_model*. The function *store_model* takes, besides the above-mentioned inputs, also the trained model, a description String of the model and the number of training samples and uses these inputs to

1. Store the trained model in binary format in a specified folder,
2. Write a database entry in the model database *trained_models.*

First, a unique name for the pickled model is created, encoding the *usecase*, *foodtype* and *sensor* or combination of *sensors* used. Then the model is binarized and saved to disk. In a third step, a database entry is created with the following fields:

- *UseCase* : String holding the use case

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

- *foodType* : String holding the food type
- *sensors* : list of Strings with the used sensors
- *targets* : list of Strings with the target fields
- *constraints* : dictionary of constraints used to query the training data
- *nTrainingSamples* : int, number of training samples used to train the model
- *trainedModel* : String with the full path to the pickled model
- *date* : a python *datetime* object encoding the time of creation of the model
- *modelType* : String, either *classifier* or *regressor*, indicating the type of model
- *description* : String with a verbal description of the model

Therefore, the training of machine learning models in the PhasmaFOOD cloud platform is done by calling from the terminal a python script that takes no input parameters and performs the following steps:

1. Load a training dataset for a specific combination of *usecase*, *foodtype*, *sensor* and *target* variables
2. Set up the model as a *sklearn.pipeline.Pipeline* object, optionally with hyper-parameter tuning using *sklearn.model_selection.GridSearchCV.*
3. Call the model's *fit* method with the loaded training data.
4. *Pickle* the trained model and write a database entry in the *trained_models* database.

The trained model can hence be found in the *trained_models* database and thus be loaded and used in the PhasmaFOOD software. The query_models routine is used to search the trained_models collection for finding all trained models matching its input parameters and returns a list of dictionaries with the found database entries. Input parameters are:

- A database connection *client*
- dataset parameters *usecase*, *foodtype*, *sensors*, *targets* and additional *constraints*

Once a model is selected, it can be loaded by calling *load_model* with the path to the pickled model. The dataset parameters can also be used to load the corresponding test data, again using the *load_dataset* routine and apply various evaluation metrics. The *scikit-learn* library provides a variety of the most common metrics for assessing the quality of classification and regression models in the package *sklearn.metrics*.

4.3 APIs to 3rd party databases

The PhasmaFOOD cloud platform will enable integration of 3rd party APIs which may provide complementary data and information to the data sets collected for execution of project use cases. At this point the focus resides on utilizing the available data sets from the PhasmaFOOD laboratory experiments as training and testing data sets. Although complementary data sets of spectroscopy measurements that correspond to measurements and sensor combination obtained within the project use cases is difficult to find, external to PhasmaFOOD data sets may be considered if identified and found to be fit for testing and further validation of the system.

Regarding the integration of relevant regulatory data sets for project use cases, it is worth mentioning that the acceptable and allowed levels of contaminants are currently introduced to the system according to the information provided in D1.1 [12]. This allows to put conducted analysis and measurements into regulatory context (i.e. allowed mycotoxin level in certain food types). Identified 3rd party sources of information presented in the D2.1 [1] provide adequate regulatory information to enable classification of samples. Although it can be easily supported, at this point, an automated procedure for introducing such information from an API is not available. This is mainly due to the fact that while there exist several sources that provide such information, that information is found in documents and/or webpages but unfortunately not via an API that can be used by programmers.

The PhasmaFOOD cloud platform will host open API for 3rd party developers and system integrators to be able to connect to the cloud services and develop their own mobile and web applications for food analysis. This API will be based on the cloud-mobile API described in section 3. Interactive (Swagger – OpenAPI) documentation will be provided with the link on the web dashboard and project website under development wiki page.

4.4 Matching requirements to cloud platform development

Ref. number	Description	Cloud platform development
DATA-3	A data model MUST be defined to specify how data (measurement data and corresponding metadata e.g. parameter name, timestamp, etc.) is stored and read in PhasmaFOOD system.	A data model has been implemented. More details in Section 4.1.2 and the JSON format is found in Appendix 9.1.
DATA-DB-1	The PhasmaFOOD cloud platform database MUST be able to store all measurements data coming from the portable scanner and through the PhasmaFOOD mobile app.	A cloud platform hosting PhasmaFOOD cloud database with all use case datasets is already implemented and it provides services for the mobile application and the embedded

		system.
DATA-DB-2	The PhasmaFOOD cloud platform database MUST be configured to efficiently support creation of training and test data sets for data analysis and machine learning models;	This feature is supported.
DATA-DB-3	The PhasmaFOOD cloud database SHOULD be available for 3rd party developer either as downloadable dataset or through well-defined API.	This feature is supported.
DATA-DB-4	The PhasmaFOOD cloud platform database MUST be modeled to support efficient writing, querying and archiving of measurement data and analysis results.	Part of the key functionalities implemented are to host the PhasmaFOOD cloud database and data warehouse for storing use case data sets as well as user profiles, trained ML models, decision making history and regulatory parameters.
SW-CLOUD-1	The PhasmaFOOD cloud platform MUST host trained decision making machine learning algorithms.	One of the key functionalities implemented is to host trained ML models.
SW-CLOUD-3	The PhasmaFOOD cloud platform MUST perform offline training for machine learning models based on collected data sets.	One of the key functionalities implemented is to host trained ML models based on collected data sets.
SW-CLOUD-4	The PhasmaFOOD cloud platform SHOULD enable utilization of 3rd party data sets and services providing additional data sets, context to collected data and regulatory constraints.	At this point the focus resides on utilizing the available data sets from the PhasmaFOOD laboratory experiments as training and testing data sets.
SW-CLOUD-7	The PhasmaFOOD cloud platform MUST provide API handlers for communicating with the PhasmaFOOD mobile app.	These functionalities are implemented.
SW-CLOUD-8	The PhasmaFOOD cloud platform SHOULD host a web based dashboard for managing the data sets and measurements.	
SW-CLOUD-10	The PhasmaFOOD cloud platform SHOULD host procedures for registration and authorization of individual PhasmaFOOD sensing devices.	There is a user registration mechanism in place, which is performed through the PhasmaFOOD web dashboard

SW-CLOUD-11	The PhasmaFOOD cloud platform SHOULD host procedures for end user registration, authentication and authorization.	and/or PhasmaFOOD mobile application. There is also a device registration and authorization mechanism in place.
SW-DASHB-1	The PhasmaFOOD cloud platform SHOULD provide web dashboards for system monitoring, calibration, configuration and protocol.	Web dashboard development is work in progress and will be reported and described in detail in the forthcoming deliverables D6.1 and D4.2.

5 Security and privacy

5.1 User registration and authorization

There are two types of users: administrators and end users. End users can be registered as expert or non-expert users. Based on these roles and account types, different privileges are granted (see the section 4.3). The user registration is performed through the PhasmaFOOD web dashboard and/or PhasmaFOOD mobile application. The registration form asks for a user's first and last name, email, company (optional – dropdown menu with short names of project partners) and password with strength indication and retyping. Once all information is provided, the MailGun service is used to send an email verification message to the provided email address. The message will contain a verification URL, which will automatically finalize the user registration upon opening.

The registration process utilizes JWT tokens (RFC 7519) as means for authorization and authentication of end users. The JWT token authorization mechanism example is depicted in Figure 22. In the PhasmaFOOD solution the JWT tokens are built using email and Django secret key.

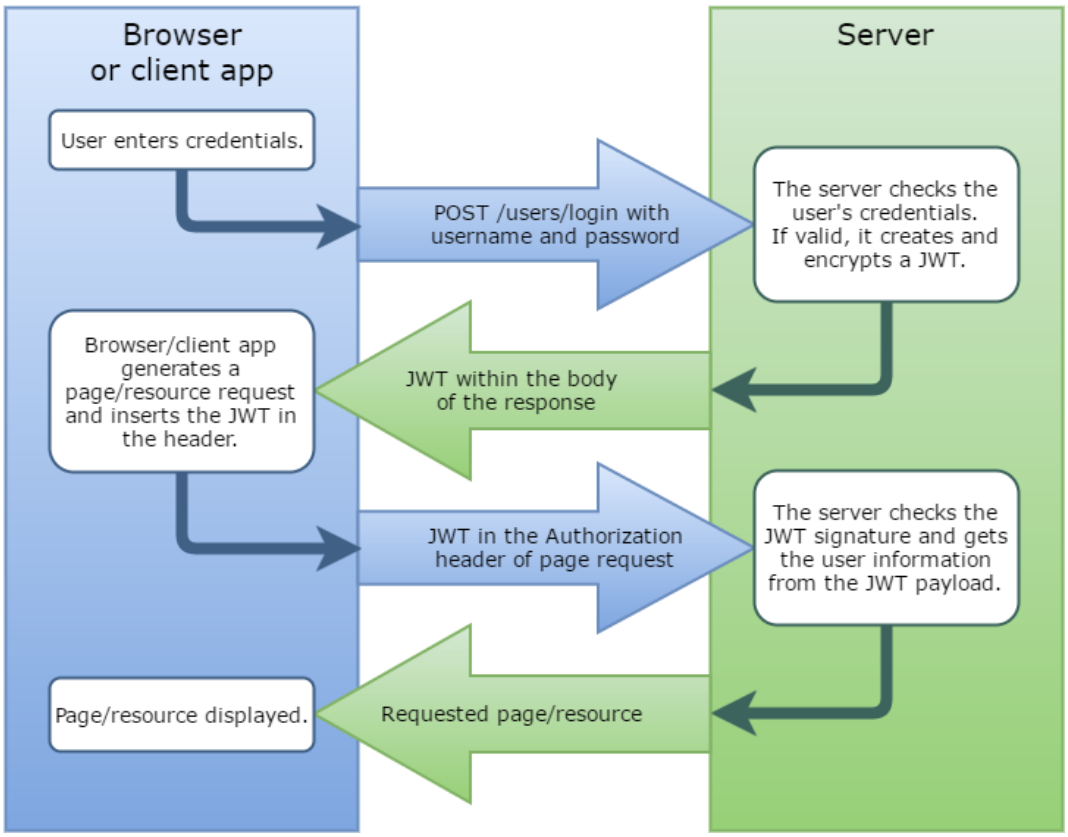

Figure 22 - JWT token based authentication and authorization

The JWT tokens secure all message exchange on cloud APIs towards outside platform elements – mobile application and web dashboard.

5.2 Device registration and authorization

Every device (smart sensing device and end user smartphones) needs to be registered on the PhasmaFOOD cloud platform before it is used. This process is handled through the PhasmaFOOD mobile application. The application reports the WiFi MAC address of the hosting smartphone and Bluetooth UUID of the connected smart sensing device. For the final prototype, only selected Bluetooth UUIDs will be allowed for registration (preregistered UUIDs of produced smart sensing devices).

5.3 Trusted data sources

The first PhasmaFOOD software platform prototype targets internal usage of the solution within the project consortium. All project partners will be considered as trusted data sources when collecting new measurements and data sets. Partners will be responsible for validating and managing their datasets hosted on the cloud platform.

5.4 Privacy protection

In the first system prototype we have implemented data encryption procedures for personal data and login parameters provided by the end user during registration. Data encryption utilizes hash functions with Password-Based Key Derivation Function 2 PBKDF2-HMAC-SHA256-based password hashing [13]. The system does not collect other personal information like IP address, location, smartphone characteristics etc.

Since user profiles are related to a measurement history, there is a risk of privacy breach with respect to analyzed food. This does not translate to private information under standard privacy protection frameworks, but the consortium plans to address this potential privacy breach risk. We will focus on advanced privacy protection policies during the final prototype development. Also, the Terms and Conditions and Privacy Protection Policy documents are in preparation for the PhasmaFOOD solution. These will be offered for reading to end user upon registration and they will need to accept the terms before being able to register.

5.5 Matching requirements to data security and privacy

Ref. number	Description	Data security and privacy
SW-SEC-1	Measurement results SHOULD be restricted to authorized users.	There is a user registration mechanism in place, which is performed through the PhasmaFOOD web dashboard and/or PhasmaFOOD mobile application. There is also a device registration and authorization mechanism in place.
SW-SEC-2	In the case end-user (food consumers) testing will be performed, PhasmaFOOD users MUST pre-register and consent using the system.	This requirement refers to the user and device registration mechanisms, which are implemented.
SW-SEC-3	The users SHOULD have roles associated with them to determine the level of access that a user will be given and what they will be permitted to do.	End users can be registered as expert or non-expert users. Based on these roles and account types different privileges are granted (see the section 4.3).
SW-SEC-4	Access to raw reference data MUST take place only through the partners' involvement in the performance of the laboratory measurements.	This is only feasible, when registering as an expert user.
SW-SEC-5	In the cases of trend analytics, anonymization methods SHOULD be applied as part of the built-in cloud	We have implemented data encryption procedures for personal data and login

	platform features.	parameters provided by the end user during registration. We will focus on advanced privacy protection policies during the final prototype development.
SW-SEC-6	Privacy by design requirements SHOULD be followed during development of the PhasmaFOOD software components.	
SW-SEC-7	Encryption mechanisms MAY be integrated in the PhasmaFOOD system in order to ensure data privacy policies.	

6 Software Development strategy

6.1 Languages, tools and frameworks used for development

Table 3 - Development languages, tools and frameworks used for implementation of the first prototype of the PhasmaFOOD SW platform

PhasmaFOOD SW architecture component	Selected development language, tools and frameworks
Embedded SW	• C for kernel functions and drivers • C++ for object oriented procedures
Android mobile app	• Java for native development of mobile application functionalities • Android SDK for standard Android system elements and functionalities (public classes)
Cloud backend	• Python 3.6.3 for custom backend services • Django web framework 2.0
Web dashboard	• Angular 5 for frontend framework • HTML 5 and SCSS for UI • NGINX web server for hosting the web dashboard
Cloud DB	• Operational DB – PostgreSQL which is standard DB for Django framework • Cloud database for PhasmaFOOD data sets - MongoDB
Cloud machine learning framework	• Python 3.x
APIs and notifications	• Django REST Framework for cloud-mobile, web dashboard and internal APIs • JWT token framework for authorization • Firebase Cloud Messaging service for push notifications • Swagger OpenAPI – framework for dynamic generation of interactive documentation for RESTful

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

	APIs
Data analysis algorithms development	• All data analysis algorithms in the PhasmaFOOD software solution are implemented in Python using NumPy, SciPy and scikit-learn. • Statistical toolbox of MATLAB (commercial) for assistance in the implementation of PLS and PCA and Linear Classification model. In this context, exploiting several predefined functions the development time of new data analysis algorithms can be dramatically reduced. Free MATLAB toolbox developed by the "Milano Chemometrics group"[1]. Note that Matlab is not employed in the PhasmaFOOD software solution, it is only used as a supporting tool.

More information on the cloud related software development is found below.

The development of the software related to the organization and cleansing of the measurements data from various laboratories, together with their preprocessing and averaging is carried out in Pycharm IDE Community Edition [14] using Python v3.6 [9]. The same tools and programming language are used for the implementation of the data parsing/loading software responsible for loading the data to the cloud database, as well as for the software capable of extracting the test data from the database into .csv files. For various necessary numerical computations, the NumPy [15] Python library was used (v1.14). For the tools necessary to work with MongoDB [16], the PyMongo [10] library was used (v3.6.0). The produced code is maintained in both a private git repository, which is hosted in Gitlab [17] and a similar private git repository provided by Fraunhofer University for the project needs. The code is organized in three branches: A master branch (master), a development branch (dev) and a development branch with an alternative approach with respect to the splitting of the data into test and training sets (devAltSplit).

The cloud database is an instance of a MongoDB that is hosted on a Google Cloud Compute Engine virtual machine. The virtual machine is an Ubuntu Linux [18] host (running version 1604-xenial) with MongoDB server version (v3.6.3). In order to allow a web interface to folders inside the VM, it was chosen to mount a Google Cloud Storage Bucket resource [19]. This allows for easier uploading and downloading of files by using drag and drop and in a similar fashion to the known online drives. The above setup represents what could be considered as the experimental production version of the cloud database, while an identical setup exists at another virtual machine representing a development environment that allows experimentation without affecting the production deployment. It is worth mentioning that the above setup can easily scale out across several hosts to accommodate large amounts of data by exploiting the sharing

[1] http://michem.disat.unimib.it/chm/download/classificationinfo.htm

Project Title: PhasmaFOOD
Contract No. 732541
Project Coordinator: INTRASOFT International S.A.

features of MongoDB. Furthermore, the replication features available can ensure that failures will not compromise data availability.

6.2 Open source libraries

Table 4 - Open source libraries for realization of the first PhasmaFOOD software platform prototype

Data analysis component	Open source library
Embedded system	• Rapidjson for json messages. http://rapidjson.org/ • Bluez stack for bluetooth comms. http://www.bluez.org/ • Debian 9 (includes SPI, serial communications, OpenCV and other open source libraries)
Android mobile app	• Dagger2 - library for powerful dependency injection by creating tree of objects which can be easily used without complex objects initialization. • MVVM (model view view-model) - architecture which allows efficient layers decoupling, simple testing and seamless app upgrading with new features. • RxJava is used for creating reactive streams and advanced data processing and execution. • For discovering, standard Bluetooth library is used. It is wrapped with additional layer that creates reactive streams (RxBluetooth) for sending data, receiving data and observing Bluetooth states (discovering, bonding, connection etc.). • MPAndroidChart is used for plotting charts.
Cloud backend services	• ZeroMQ library for asynchronous messaging and PULL-PUSH design pattern. • For various necessary numerical computations, the NumPy Python library was used (v1.14). For the tools necessary to work with MongoDB, the PyMongo library was the API of choice (v3.6.0). In cases, where archiving was necessary Tar (GNU tar v1.28) [20] and Zip/Unzip (Debian Zip Unzip v6.0) [21]. • For database access the PyMongo v3.6.0 library is used. Computations are done using NumPy v1.14.2 and SciPy v1.0.1
Cloud based data analysis and machine learning	• Scikit-Learn – Python library for building data analysis pipelines and training classification and regression ML models. • For visualization purposes matplotlib v2.2.2, seaborn v0.8.1

	and pandas v0.22.0– plotting charts based on measurement data.
Cloud data analysis algorithms	• TensorFlow - It was originally developed by the Google Brain Team to treat perceptual and language understanding tasks. It can also be used to do research on machine learning and deep neural networks. It has a Python-based interface. https://www.tensorflow.org

6.3 Validation plan

6.3.1 Training and test data sets

The process of introducing data sets originating from PhasmaFOOD laboratory experiments conducted within the framework of WP3 included, among other tasks, the organization of the available information in a consistent folder structure. Furthermore, the data where cleansed whenever necessary and it was ensured that relative contamination information was matched with the available sample measurements' .csv files. The measurement files number in each sensor case was confirmed and it was ensured that the relative reference files were available, wherever such files were applicable. The code developed specifically for the purpose of loading a big number of measurements in batch mode fulfilled the task of combining all available information from measurements files into sample entries in the cloud database (according to the data model seen in Appendix A).

The available samples were divided into training and test portions. This was achieved by employing a random splitting approach, based on a configurable percentage value that splits the available data in test and training sets, when they are loaded in the cloud database. While for prototype tests a 70/30 ratio was used resulting in 70% training data and 30% test data, choosing other split ratios can easily be achieved by the implemented tools. The training portion was used in the PhasmaFOOD cloud platform to train the machine learning models for classification and regression, which are part of the decision-making mechanisms for project use cases. The test/validation portion of these data sets were used to emulate real measurements, coming from the integrated PhasmaFOOD smart sensing device.

For the purpose of extracting the test data in a suitable format and in order to ensure in the process that training data will never be used in subsequent tests, a set of scripts was developed that serves the exact purpose of extracting the test data from the cloud database (30% of the data sets) and generating files identical to the ones that would have been produced by the actual sensor hardware, for all sensor types. In addition to the test data, the extraction scripts also provide the classification information of the samples in a suitable format together with relative information (contamination levels, aflatoxins, microbiological data), providing adequate information to simulate a measurement and verify that the correct classification response was subsequently received. It is also worth mentioning that specifically for the cases, were averaging and/or preprocessing is to be performed on the embedded device (e.g. average

across measurements, Dark subtraction and White division), the averaged and pre-processed values are also calculated and available in the cloud database. This provides a mechanism to further ensure, if necessary, the fact that pre-processed and averaged values originating from the embedded device are matching to the ones that were calculated by a different implementation on a different platform.

6.3.2 Training of machine learning models on the cloud platform

The detailed procedure of training machine learning models on the cloud platform is described in Section 4.2. Furthermore, the data analysis algorithms recommended by the use case leaders and described in Deliverable Report D3.3 will be implemented in the same fashion. A set of concurrent algorithms will be trained and tested as well in order to offer a variety of models and hence ensure the quality of the obtained predictions. A web application to build new machine learning models and test them on several datasets is currently under development. This will enable all researchers in the PhasmaFOOD project to try out new custom algorithms and use them inside the PhasmaFOOD software solution.

6.3.3 Testing of communication flow

The communication flows – APIs are tested between the participating software platform layers.

- Embedded–mobile API was tested between the emulated embedded system running on the BeagleBone Black microprocessor board and the Android application. Embedded development team had custom Android application for testing the Bluetooth communication mechanism and message exchange. Next, as the PhasmaFOOD android application was developed to support measurement configuration and communication with the cloud platform, testing of the communication API was performed through joint efforts of the mobile app and embedded system development teams. During the testing the API was optimized, and the utilized data models and instructions parameters were aligned with the rest of the system (cloud platform and DB).
- Cloud-mobile API as a first step in development included user registration and authentication mechanism based on the JWT tokens and the Django REST framework. This was necessary to complete user profiles and device profiles models in the operational DB. Next, the actual measurement data and data analysis results exchange messages are implemented and tested. Again, the REST API procedures are optimized and connected with the cloud platform data flow processes. The push notifications API is also implemented to enable mobile application to receive analysis results in a session-less manner.
- Internal cloud APIs were implemented with the Django REST framework and enable direct communication of the cloud platform processes. These APIs are tested together with development and optimization of the cloud procedures.

6.3.4 Testing of communication interfaces

The first proof of concept software platform was built to test the communication flows and the decision making algorithms for selected use cases and sample types. The data analysis pipelines are optimized for selected use cases and scenarios. During testing of the rule engine, it was noticed that sequential execution of the data analysis pipelines will slow down system response and end users would have to wait longer to receive the food analysis results. This would be more evident with increasing number of users and smart sensing devices being active at the same time. We have decided to utilize multithreading mechanism and enable the rule engine to instantiate new processing threads and execute selected data analysis and ML pipelines base on input parameters. The ZeroMQ library is used for managing message queues and organizing execution of analysis tasks with controllable execution priority policies.

The reactive decision making procedures are tested and validated in the first proof of concept implementation, where the measurement process was emulated on the embedded device. First, we have divided existing data sets into training and test portions (70%-30%). The test portions were stored in the embedded system storage. All data set entries have labels, which classify collected measurements based on contamination level and/or microbiological values. These are key metrics used by food experts to classify samples and ensure that all relevant classes are represented in the data sets. The machine learning models for classification and regression are trained with these metrics as classes and value levels. Based on the measurement configuration sent from the mobile application, the embedded system will choose randomly one measurement datum (dataset entry) from the corresponding dataset and report it back to the mobile application together with the assigned measurement label. The mobile application will save the received correct label from the test dataset entry and send the measurement data to the cloud platform for analysis. The reactive decision making employs the appropriate data analysis and classification/regression pipeline, based on the measurement configuration. The analysis result is a set of labels obtained from analyzing and classifying measurement data for NIR and UV-VIS spectrometers. The results are sent to the mobile application and end user is presented with the result/label obtained with cloud based data analysis and real/correct label obtained from the test dataset entry, which was selected to emulate a real measurement. In this way project partners were able to validate the performance of implemented ML models and quality of the available dataset for training high performance classification and regression models.

6.3.5 Testing of decision making algorithms

The performance of decision making algorithms is tested before deployment using the dedicated testdata for each dataset. For classification tasks, the function *sklearn.metrics.classification_report()* is then applied to the ground truth and the predicted labels to obtain precision, recall and f-score of the classification. A confusion matrix is calculated with the function *sklearn.metrics.confusion_matrix()*, showing the number of correctly classified samples on the main diagonal and the number samples from class *i*, wrongly classified as belonging to class *j* in the entry *(i,j)*. For regression tasks, the mean squared error,

the mean and median absolute errors and the explained variance are calculated using the respective functions *sklearn.metrics.mean_squared_error(), sklearn.metrics.mean_absolute_error(), sklearn.metrics.median_absolute_error() and sklearn.metrics.explained_variance()*. A standardized test suite is currently being developed and will be part of the web dashboard.

7 Conclusion and next steps

In this deliverable the PhasmaFOOD consortium presented the implementation details of the software components of the PhasmaFOOD system. The software implementation has followed the project's functional and system specifications and requirements. An overview of the software architecture, the embedded software, the mobile and cloud platform is described. The deliverable presented a set of approaches for implementation of data analysis and machine learning processes, necessary for realization of the project use cases, as part of the overall software architecture. The PhasmaFOOD mobile application and the main building components of the PhasmaFOOD cloud platform are also defined. The cloud database design and high level sequence diagrams for all APIs are presented. Also, user interfaces are presented as well as usage UML diagrams for the mobile application and web dashboard featured on the PhasmaFOOD cloud platform. Finally, the deliverable reports on the main security and privacy protection procedures that are implemented on the software architecture. At the end, the document gives an overview of the software development strategy, techniques and tools that have been utilized in realization of the PhasmaFOOD software architecture and its components.

As the next steps, this deliverable will be utilized by WP6 for the consortium to deliver the first integrated functional PhasmaFOOD prototype. Task 6.1 will receive the D4.1 deliverable as the main input providing the software components (embedded, mobile and cloud) to be integrated in the PhasmaFOOD system.

8 References

[1] PhasmaFOOD, D2.1 - Software specifications for PhasmaFOOD smart food analysis system, July 2017.

[2] *PhasmaFOOD, "D5.3 - Implementation of embedded software for PhasmaFOOD smart food analysis platform", April 2018.*

[3] *PhasmaFOOD, "D3.1 - Feasibility results and use case benchmarking-v1," Sept 2017..*

[4] *D.L. Massart, B.G. Vandegiste, S.N. Deming, Y. Michotte, L. Kaufmann; Data handling in science and technology vol.2: Chemometrics: a textbook, Elsevier (Amsterdam, The Netherlands), 1988.*

[5] *Geladi, P., & Kowlaski B. (1986). Partial least square regression: A tutorial. Analytica Chemica Acta, 35, 1-17.*

[6] *Martens, H, & Naes, T. (1989). Multivariate Calibration. London: Wiley..*

[7] *Richard O. Duda, Peter E. Hart, and David E. Stork. Pattern Classification. Wiley, New York, second edition, 2001.*

[8] "JSON Schema | The home of JSON Schema," JSON Schema org, [Online]. Available: http://json-schema.org/. [Accessed 01 05 2018].

[9] "Welcome to Python," Python Software Foundation, [Online]. Available: https://www.python.org/. [Accessed 01 05 2018].

[10] "PyMongo - MongoDB API," MongoDB, Inc, [Online]. Available: https://api.mongodb.com/python/current/. [Accessed 01 05 2018].

[11] "Scikit-learn: Machine Learning in Python," [Online]. Available: http://scikit-learn.org/stable/index.html. [Accessed May 2018].

[12] PhasmaFOOD, D1.1 - Use cases description, May 2017.

[13] *http://passlib.readthedocs.io/en/stable/lib/passlib.hash.pbkdf2_digest.html#passlib.hash.pbkdf2_sha256.*

[14] JetBrains, "PyCharm: Python IDE for Professional Developers by JetBrains," JetBrains.com, [Online]. Available: https://www.jetbrains.com/pycharm/. [Accessed 01 05 2018].

[15] "NumPy," NumPy.org, [Online]. Available: www.numpy.org. [Accessed 01 05 2018].

[16] "MongoDB for GIANT Ideas | MongoDB," MongoDB, Inc, [Online]. Available: https://www.mongodb.com/. [Accessed 01 05 2018].

[17] "The only single product for the complete DevOps lifecycle - GitLab," GitLab Inc, [Online]. Available: https://gitlab.com/. [Accessed 01 05 2018].

[18] "Ubuntu 16.04.4 LTS (Xenial Xerus)," Ubuntu, [Online]. Available: releases.ubuntu.com/16.04/. [Accessed 01 05 2018].

[19] "Buckets | Cloud Storage Documentation | Google Cloud," Google Inc, [Online]. Available: https://cloud.google.com/storage/docs/json_api/v1/buckets. [Accessed 01 05 2018].

[20] "Tar - GNU Project - Free Software Foundation," GNU Operating System FSF, [Online]. Available: https://www.gnu.org/software/tar/. [Accessed 01 05 2018].

[21] "Debian Details of Package," Debian Project SPI Inc, [Online]. Available: https://packages.debian.org/stretch/unzip. [Accessed 01 05 2018].

[22] *Martens, H, & Naes, T. (1989). Multivariate Calibration. London: Wiley..*

Appendix A

8.1 Data Model (JSON Schema Format)

```
"sampleID":{ "type":"string" },
"laboratory":{ "type":"string" },
"userID":{ "type":"string" },
"deviceID":{ "type":"string" },
"foodType":{ "type":"string" },
"useCase":{ "type":"string" },
"replicate":{ "type":"int" },
"temperature":{ "type":"int" },
"temperatureProfile":{ "type":"string" },
"tempExposureHours":{ "type":"int" },
"packageType":{ "type":"string" },
"sampleStatus":{ "type":"string" },
"measurementFreq":{ "type":"int" },
"tvc":{ "type":"float" },
"dateTime":{ "type":"date" },
"granularity":{ "type":"string" },
"contamination":{ "type":"int" },

"aflatoxin":{ "type":"object", "properties":{
   "name":{ "type":"string" },
   "quantification":{ "type":"float" },
   "unit":{ "type":"string" }}},

"customType1":{ "type":"string" },
"customType2":{ "type":"string" },
"mainCat":{ "type":"string" },
"cat":{ "type":"string" },
"adul":{ "type":"int" },

"microbiologicalMeasurement":{ "type":"object", "properties":{
   "microbiologicalUnit":{ "type":"string" },
   "microbiologicalValue":{ "type":"float" }}},

"experiment":{ "type":"object", "properties":{

   "FLUO":{ "type":"object", "properties":{

      "wavelengths":{ "type":"array", "items":{
         "type":"float" }},

      "measurements":{ "type":"object", "properties":{

         "trainData":{ "type":"array", "items":{
            "type":"object", "properties":{

               "columnData":{ "type":"object", "properties":{
```

```
            "measurements":{ "type":"array", "items":{
              "type":"array", "items":{
                "type":"float" }}},

            "dark":{ "type":"array", "items":{
              "type":"array", "items":{
                "type":"float" }}},

            "ref":{ "type":"array", "items":{
              "type":"array", "items":{
                "type":"float" }}}}},

          "averageData":{ "type":"object", "properties":{

            "measurements":{ "type":"array", "items":{
              "type":"float" }},

            "dark":{ "type":"array", "items":{
              "type":"float" }},

            "ref":{ "type":"array", "items":{
              "type":"float" }}}}}},

        "testData":{ "type":"array", "items":{
          "type":"object", "properties":{

            "columnData":{ "type":"object", "properties":{

              "measurements":{ "type":"array", "items":{
                "type":"array", "items":{
                  "type":"float" }}},

              "dark":{ "type":"array", "items":{
                "type":"array", "items":{
                  "type":"float" }}},

              "ref":{ "type":"array", "items":{
                "type":"array", "items":{
                  "type":"float" }}}},

            "averageData":{ "type":"object", "properties":{

              "measurements":{ "type":"array", "items":{
                "type":"float" }},

              "dark":{ "type":"array", "items":{
                "type":"float" }},

              "ref":{ "type":"array", "items":{
                "type":"float" }}}}}}},

      "overallAverages":{ "type":"array", "items":{
        "type":"float" }}}},
```

```
"VIS":{ "type":"object", "properties":{

  "wavelengths":{ "type":"array", "items":{
    "type":"float" }},

  "measurements":{ "type":"object", "properties":{

    "trainData":{ "type":"array", "items":{
      "type":"object",  "properties":{

        "columnData":{ "type":"object", "properties":{

          "measurements":{ "type":"array", "items":{
            "type":"array", "items":{
              "type":"float" }}},

          "dark":{ "type":"array", "items":{
            "type":"array", "items":{
              "type":"float" }}},

          "ref":{ "type":"array", "items":{
            "type":"array", "items":{
              "type":"float" }}}}},

        "averageData":{ "type":"object", "properties":{

          "measurements":{ "type":"array", "items":{
            "type":"float" }},

          "dark":{ "type":"array", "items":{
            "type":"float" }},

          "ref":{ "type":"array", "items":{
            "type":"float" }}}}}},

    "testData":{ "type":"array", "items":{
      "type":"object", "properties":{

        "columnData":{ "type":"object", "properties":{

          "measurements":{ "type":"array", "items":{
            "type":"array", "items":{
              "type":"float" }}},

          "dark":{ "type":"array", "items":{
            "type":"array", "items":{
              "type":"float" }}},

          "ref":{ "type":"array", "items":{
            "type":"array", "items":{
              "type":"float" }}}}},
```

```
            "averageData":{ "type":"object", "properties":{
              "measurements":{ "type":"array", "items":{
                "type":"float" }},

              "dark":{ "type":"array", "items":{
                "type":"float" }},

              "ref":{ "type":"array", "items":{
                "type":"float" }}}}}}},

        "overallAverages":{ "type":"object", "properties":{

          "averageDark":{ "type":"array", "items":{
            "type":"float" }},

          "averageRef":{ "type":"array", "items":{
            "type":"float" }},

          "averageMeasurement":{ "type":"array", "items":{
            "type":"float" }}}},

        "preprocessedValues":{ "type":"array", "items":{
          "type":"float" }}}}}},

    "NIR":{ "type":"object", "properties":{

      "wavenumbers":{ "type":"array", "items":{
        "type":"int" }},

      "trainData":{ "type":"array", "items":{
        "type":"object", "properties":{

          "intensity":{ "type":"array", "items":{
            "type":"float" }},

          "absorbance":{ "type":"array", "items":{
            "type":"float" }}}}},

      "testData":{ "type":"array", "items":{
        "type":"object", "properties":{

          "intensity":{ "type":"array", "items":{
            "type":"float" }},

          "absorbance":{ "type":"array", "items":{
            "type":"float" }}}}},

      "averageIntensity":{ "type":"array", "items":{
        "type":"float" }},

      "averageAbsorbance":{ "type":"array", "items":{
        "type":"float" }},
```

"whiteReference":{ "type":"array", "items":{
"type":"float" }},

"darkIntensity":{ "type":"float" }}},

"MSI_IMG":{ "type":"object", "properties":{

"imageName":{ "type":"string" },

"image":"image"}}}}